Database Management

CS 4342 / CS5342

Spring 2022 Semester

Final Project

Front Desk

The Miners Team

Christian Gomez,

Garrett Jones,

Miguel Rodarte,

Alan Verdin

Table of Contents

1. Scope 3
2. Requirements and Assumptions 4
3. E/R Diagram 5
4. Relational Model 6
5. Normalization Process 7
6. Normalized Schema 8
7. MySQL Server 9
8. Database Records 12
9. SQL Queries 16
10. Graphical User Interface 20
11. ISS Webserver and GUI part 1 30
12. Views 34
13. Procedures 34
14. Reports 34
15. Requirements Tracing 36
16. Graphical User Interface Part 2 42
17. ISS Webserver and GUI part 2 50
18. References 51

60

Appendix A

1. SCOPE

We have been hired by the University of Texas at El Paso to create a system that logs requests to the front desk. The system will allow visitors such as students, staff, faculty, and guests to make at least one or multiple request(s). The requests can be made by email, phone, or walk-in. Email or phone requests must be entered into the system by an assistant from the front desk, otherwise the visitor will be able to interact directly with the system. Staff members will be responsible for creating and filling in the request in case where the request is type email or phone. The system will ask the visitor to input data such as name, type of visitor and a description of the problem. The date and time can be assigned by the system. The visitor can have/enter more than one phone number. If the requester information is removed, the requests made by that requester must also be removed. Once the request is completed then it will be sent to the service department.

A friendly user interface needs to be made to accompany the system. It will be friendly for the user by providing exact input locations with adequate labels to prevent confusion or input errors. The interface will allow the staff from the front desk or visitor to properly input the visitor's data from the request in a fast paced and error-free manner. Once the data is submitted to the system, assistants will be able to make a change of request status, forward the request to the designated service and receive assistance from the desired service.

The data relevant for the system ranges from the visitor's personal information such as full name, id, email, phone number and visitor type; staff, faculty, student, or guest. This will help the system properly identify the visitor. The system will take in the user's description of the problem and indicate a forward to selection. The selection will help direct the user request to the correct service, the services and their designated staff will be part of the data file. Having this information in a data file will allow the staff to add/remove services as well as staff.

The system will collect the most important data such as time/days the front desk received more requests, the number of requests attended per day, the number of visitors assisted per day or week, most common type of visitor, most common type of request, most common request category, and number of requests done by each category. Collected data will be reported per day and week.

2. REQUIREMENTS

NEW REQUIREMENTS:

- R1. The system shall record the first name, last name, email, phone number and ID of each visitor.
- R2. The system shall assign a unique ID for each request. This ID will be used to track the request much faster in the database.
- R3. The system shall allow visitors and staff to cancel a request.
- R4. The system will assign a pending status by default when a request is done by a visitor.
- R5. The system will provide a list of options to declare if the request was done by email, phone, or walk-in.
- R6. Only staff members will be able to modify the status of the request, if needed.
- R7. The system shall allow staff members to change any information from the request. For example: A wrong or bad description about the problem, incorrect or missing information.
- R8. The system shall allow assistant personnel to generate reports about the most common requests, the total count of each type of request (email, phone, walk-in), days with more visits, most common type of visitors, view of all the requests, view of all deleted requests, all request done by a type of visitor, and provide a report menu.
- R9. The system shall allow the assistant attending a request to change the request's status from pending to "in progress"," cancel", or "done".
- R10. The system will provide a user-friendly interface where the visitor can find and create a request easily. This will be implemented by adding color buttons and clear text messages specifying what that buttons does.

NEW ASSUMPTIONS:

- Visitors can make multiple requests.
- Each request will be assigned by only one staff member.
- Each request will have a unique ID number.
- Staff members or visitors can cancel the request if they want to.
- In case a staff member cancels a request, this one will provide reasons/feedback to the visitor.

Commented [JG1]: how the system will identify this? This should be an enter data provided by the person who is entering the request. Any other ideas is welcome. The designer an programmer can include a list of three options: walk-in, phone, email, and the requester or person who is creating the request can choose the correct option.

Commented [JG2]: Good.

Commented [JG3]: Add: number of request per day, per week, per period of time.

Commented [JG4R3]: Who assisted more requests per day? or per period of time?

Commented [JG5R3]: Number of request pending per period of time.

Commented [JG6]: How are you going to validate this?

3. ENTITY RELATIONSHIP DIAGRAM

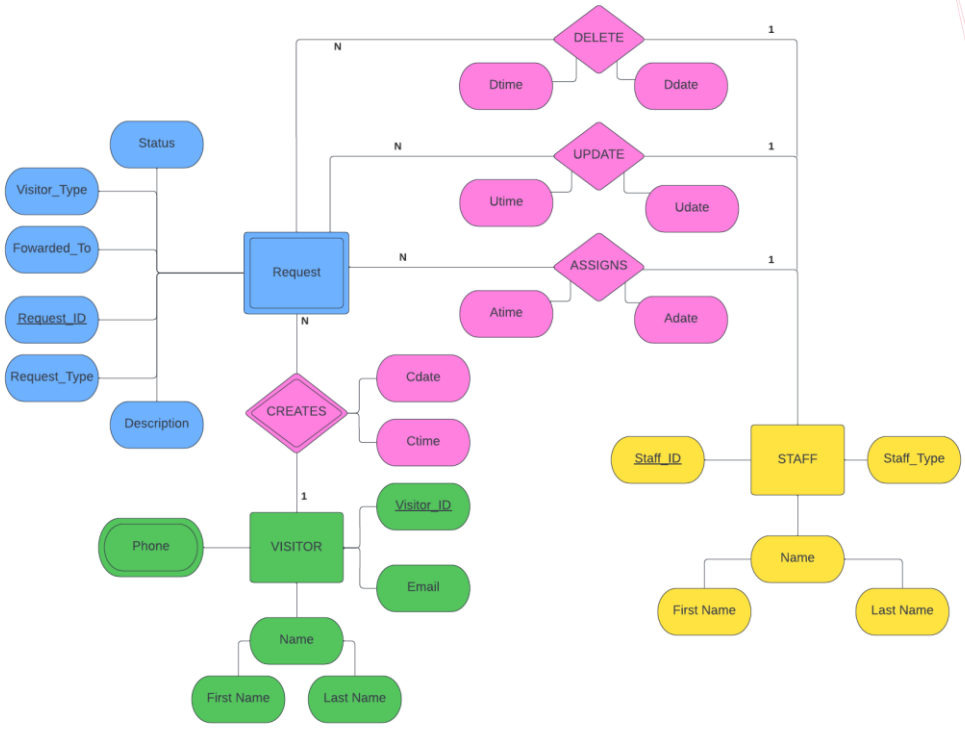


Figure 3.1: E/R Diagram

Commented [JG7]: Staff ASSIGNS request means "create" right?

Commented [JG8R7]: Assign specific date and time for each relationship type. The way you diagram show this (Pink figures on the right) is incorrect. It seems they are shared the same attributes and they are not. Each delete, update and assigns process has its specific date and time. (del_time, del_date), (update_time, update_Del), (assign_time, assign_date).

Commented [JG9R7]: Missing total participation.

Commented [JG10R7]: We said that visitor's email should be split into username and emailaccountytpe, otherwise how do you know if it is an utep or miners type?

4. RELATIONAL MODEL

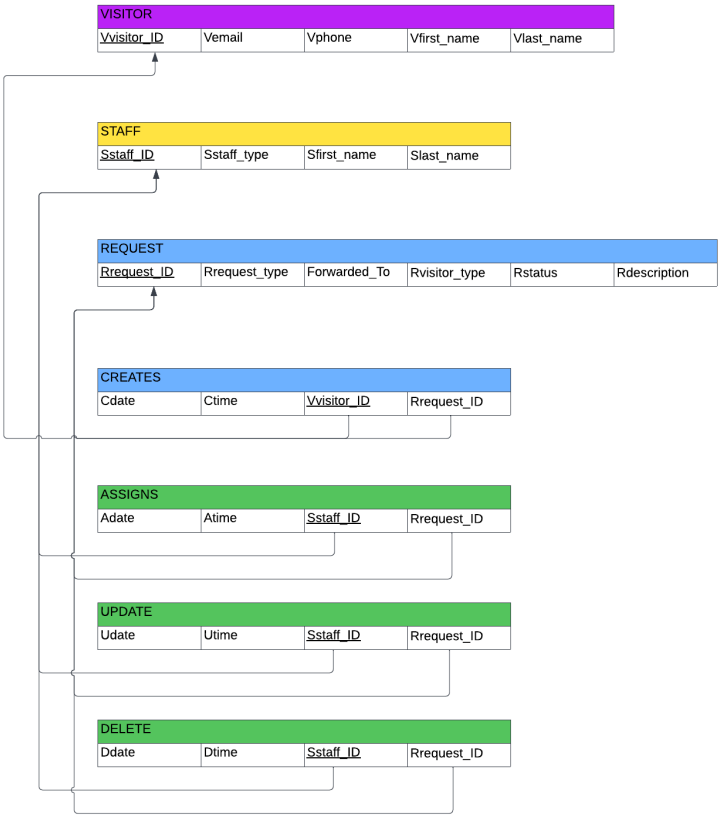


Figure 4.1: Relational Model

5. NORMALIZATION PROCESS

-Here we are just showing the process we did to get our normalized schema. If something is not clear, then we can check again here all the process. Also, here we explain for each table if it is **1NF,2NF or 3NF.**

Figure 5.1: Normalization Process

6. NORMALIZED SCHEMA

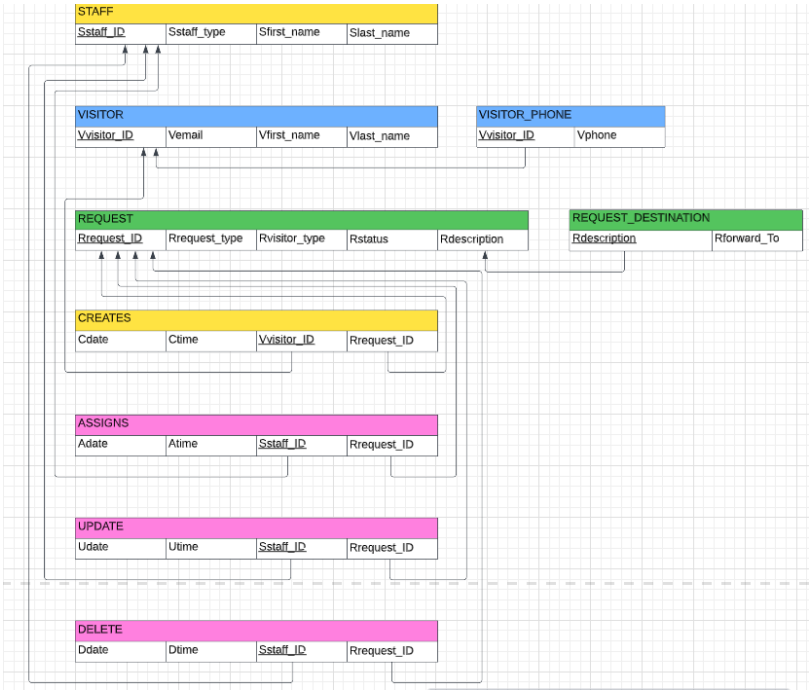


Figure 6.1: Normalization

Functional Dependencies:

FD1:{Vvisitor_ID}->{Vemail,Vfirst_name,Vlast_name,Vphone},

{Rrequest_ID}->{Rrequest_type,Rforward_to,Rvisitor_type,Rstatus,Rdescription}.

{Vvisitor_ID}->{Cdate,Ctime,Rrequest_ID}

{Sstaff_ID}->{Adate,Atime,Rrequest_ID}

FD2: NONE.

FD3: {Rforward_to}->{Rdescription}

7. MySQL SERVER

The following images show the Create statements for the relations in section 6. Assigns, updates, deletes were created by Garrett Jones, Request and Staff were created by Christian Gomez, Request_destination and creates were made by Miguel Rodarte, and Visitor and Visitor_phone were made by Alan Verdin.

Request:

```
SQL > CREATE TABLE IF NOT EXISTS Request(
   -> Rrequest_id INT NOT NULL,
   -> Rrequest_type VARCHAR(100),
   -> Rvisitor_type VARCHAR(100),
   -> Rstatus VARCHAR(100),
   -> Rdescription VARCHAR(100),
   -> PRIMARY KEY(Rrequest_id)) ENGINE=InnoDB;
```

Figure 7.1: Creation of Request Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe request;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Rrequest_id   | int(11)     | NO   | PRI | NULL    |       |
| Rrequest_type | varchar(100)| YES  |     | NULL    |       |
| Rvisitor_type | varchar(100)| YES  |     | NULL    |       |
| Rstatus       | varchar(100)| YES  |     | NULL    |       |
| Rdescription  | varchar(100)| YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Figure 7.2: Describing Request

Staff:

```
s22_mjv_team5  SQL > CREATE TABLE IF NOT EXISTS Staff(
                  -> Sstaff_id INT NOT NULL,
                  -> Sstaff_type VARCHAR(100),
                  -> Sfirst_name VARCHAR(100),
                  -> Slast_name VARCHAR(100),
                  -> PRIMARY KEY(Sstaff_id)) ENGINE=InnoDB;
```

Figure 7.3: Creation of Staff Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe staff;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Sstaff_id   | int(11)     | NO   | PRI | NULL    |       |
| Sstaff_type | varchar(100)| YES  |     | NULL    |       |
| Sfirst_name | varchar(100)| YES  |     | NULL    |       |
| Slast_name  | varchar(100)| YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

Figure 7.4: Describing Staff

Request_destination:

```
CREATE TABLE request_destination(Rdescription CHAR(20) PRIMARY KEY, Rforward to CHAR(20))Engine=InnoDB;
```

Figure 7.5: Creation of request_destination table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe request_destination;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| Rdescription | char(20) | NO   | PRI | NULL    |       |
| Rforward_to  | char(20) | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
```

Figure 7.6: Describing request destination

Creates:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE TABLE creates (Cdate CHAR(10), Ctime CHAR(15),
(Rrequest_ID) REFERENCES request(Rrequest_ID)) ENGINE=InnoDB;
Query OK, 0 rows affected (1.1432 sec)
```

Figure 7.7: Creation of creates table

```
Vvisitor_ID INT NOT NULL, Rrequest_ID INT NOT NULL, FOREIGN KEY(Vvisitor_id) REFERENCES Visitor(Vvisitor_id), FOREIGN KEY
```

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe creates;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| Cdate       | date     | YES  |     | NULL    |       |
| Ctime       | char(15) | YES  |     | NULL    |       |
| Vvisitor_ID | int(11)  | NO   | MUL | NULL    |       |
| Rrequest_id | int(11)  | NO   | MUL | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

Figure 7.8: Describing creates table

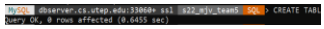
Updates:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > create table updates(
NULL, Rrequest_id int, FOREIGN KEY(Sstaff_id) REFERENCES staff(Sstaff_id), FOREIGN
) ENGINE = InnoDB;
Query OK, 0 rows affected (0.7379 sec)
```

```
(Udate DATE, Utime VARCHAR(15), Sstaff_ID INT NOT
 KEY (Rrequest_id) REFERENCES Request(Rrequest_id)
```

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe updates;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Udate       | date        | YES  |     | NULL    |       |
| Utime       | varchar(15) | YES  |     | NULL    |       |
| Sstaff_ID   | int(11)     | NO   | MUL | NULL    |       |
| Rrequest_id | int(11)     | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.0489 sec)
```

Assigns:



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > create table assigns(
NULL, Rrequest_id int, FOREIGN KEY(Sstaff_id) REFERENCES staff(Sstaff_id), FOREIGN
) ENGINE = InnoDB;
Query OK, 0 rows affected (0.9806 sec)
```

```
(Adate DATE, Atime VARCHAR(15), Sstaff_ID INT NOT
 KEY (Rrequest_id) REFERENCES Request(Rrequest_id)
```

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe assigns;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Adate       | date        | YES  |     | NULL    |       |
| Atime       | varchar(15) | YES  |     | NULL    |       |
| Sstaff_ID   | int(11)     | NO   | MUL | NULL    |       |
| Rrequest_id | int(11)     | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.0682 sec)
```

Deletes:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > create table deletes(
request_id int, FOREIGN KEY(Sstaff_id) REFERENCES staff(Sstaff_id), FOREIGN KEY (Rr
E = InnoDB;
Query OK, 0 rows affected (0.7727 sec)
```

```
 DATE, Dtime VARCHAR(15), Sstaff_ID INT NOT
Rrequest_id) REFERENCES Request(Rrequest_id)
```

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe deletes;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Ddate       | date        | YES  |     | NULL    |       |
| Dtime       | varchar(15) | YES  |     | NULL    |       |
| Sstaff_ID   | int(11)     | NO   | MUL | NULL    |       |
| Rrequest_id | int(11)     | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.0664 sec)
```

Visitor:

```
s22_mjv_team5  SQL > CREATE TABLE IF NOT EXISTS Visitor(
                  -> Vvisitor_id INT NOT NULL,
                  -> Vemail VARCHAR(100),
                  -> Vfirst_name VARCHAR(100),
                  -> Vlast_name VARCHAR(100),
                  -> PRIMARY KEY(Vvisitor_id))ENGINE=InnoDB;
```

Figure 7.19: Creation of Visitor Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe visitor;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Vvisitor_id | int(11)      | NO   | PRI | NULL    |       |
| Vemail      | varchar(100) | YES  |     | NULL    |       |
| Vfirst_name | varchar(100) | YES  |     | NULL    |       |
| Vlast_name  | varchar(100) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

Figure 7.20: Describing Visitor Table

Visitor_phone:

```
s22_mjv_team5  SQL > CREATE TABLE IF NOT EXISTS Visitor_phone(
                  -> Vvisitor_id INT NOT NULL,
                  -> Vphone VARCHAR(100),
                  -> FOREIGN KEY(Vvisitor_id)
                  -> REFERENCES Visitor(Vvisitor_id)) ENGINE=InnoDB;
```

Figure 7.21: Creation of Visitor_phone Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > describe visitor_phone;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Vvisitor_id | int(11)      | NO   | MUL | NULL    |       |
| Vphone      | varchar(100) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

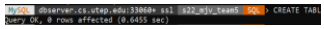
Figure 7.22: Describing Visitor_phone Table

8. DATABASE RECORDS

Figure 6 shows the database records and insert statements for assigns, updates, deletes, request, request_destination, staff, creates, visitor, and visitor_phone tables.

Assigns:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > insert into assigns values ("2022-03-05","10:00 AM MT","4001","5001"),
                                                          -> ("2022-03-25","3:00 PM MT","4001","5002"),
                                                          -> ("2022-03-10","10:50 AM MT","4002","5003"),
                                                          -> ("2022-03-02","6:00 AM MT","4003","5005"),
                                                          -> ("2022-02-12","5:15 PM MT","4003","5006"),
                                                          -> ("2021-08-28","5:57 PM MT","4002","5009");
Query OK, 6 rows affected (0.4028 sec)

Records: 6  Duplicates: 0  Warnings: 0
```



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from assigns;
+------------+-------------+-----------+-------------+
| Adate      | Atime       | Sstaff_ID | Rrequest_id |
+------------+-------------+-----------+-------------+
| 2022-03-05 | 10:00 AM MT |      4001 |        5001 |
| 2022-03-25 | 3:00 PM MT  |      4001 |        5002 |
| 2022-03-10 | 10:50 AM MT |      4002 |        5003 |
| 2022-03-02 | 6:00 AM MT  |      4003 |        5005 |
| 2022-02-12 | 5:15 PM MT  |      4003 |        5006 |
| 2021-08-28 | 5:57 PM MT  |      4002 |        5009 |
+------------+-------------+-----------+-------------+
6 rows in set (0.0520 sec)
```

Updates:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO updates VALUES ("3/05/2022","11:00 AM MT","4001","5001"),
M MT","4003","5006"), ("3/25/2022","10:20 AM MT","4002","5008");
Query OK, 5 rows affected (0.3092 sec)
```

Figure 8.3: Inserting Values to Updates Table

```
("3/10/2022","12:20 PM MT","4002","5003"), ("3/02/2022","8:00 AM MT","4003","5005"), ("2/12/2022","6:29 P
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from updates;
+-----------+-------------+-----------+-------------+
| Udate     | Utime       | Sstaff_id | Rrequest_id |
+-----------+-------------+-----------+-------------+
| 3/05/2022 | 11:00 AM MT |      4001 |        5001 |
| 3/10/2022 | 12:20 PM MT |      4002 |        5003 |
| 3/02/2022 | 8:00 AM MT  |      4003 |        5005 |
| 2/12/2022 | 6:29 PM MT  |      4003 |        5006 |
| 3/25/2022 | 10:20 AM MT |      4002 |        5008 |
+-----------+-------------+-----------+-------------+
```

Figures 8.4: Showing Updates Table content.

Deletes:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > insert into deletes values ("8/22/2021","3:25 PM MT","4001","5010"),
                                                          -> ("9/05/2021","1:05 PM MT","4001","5009");
Query OK, 2 rows affected (0.2013 sec)
```

Figures 8.5: Inserting values into deletes table.

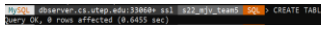
```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from deletes
                                                          -> ;
+-----------+------------+-----------+-------------+
| Ddate     | Dtime      | Sstaff_id | Rrequest_id |
+-----------+------------+-----------+-------------+
| 8/22/2021 | 3:25 PM MT |      4001 |        5010 |
| 9/05/2021 | 1:05 PM MT |      4001 |        5009 |
+-----------+------------+-----------+-------------+
2 rows in set (0.0481 sec)
```

Figures 8.6: Showing deletes table.

Staff:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO staff(Sstaff_id,Staff_type,Sfirst_name,Slast_n
ame)VALUES(4001,'Admin','Christian','Gomez'),(4002,'Customer Service','Miguel','Rodarte'),(4003,'Customer Service','Patr
ick','Star');
```

Figure 8.7: Inserting values to Staff Table



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from staff;
+-----------+-----------------+-------------+------------+
| Sstaff_id | Sstaff_type     | Sfirst_name | Slast_name |
+-----------+-----------------+-------------+------------+
|      4001 | Admin           | Christian   | Gomez      |
|      4002 | Customer Service | Miguel     | Rodarte    |
|      4003 | Customer Service | Patrick    | Star       |
+-----------+-----------------+-------------+------------+
3 rows in set (0.0430 sec)
```

Figure 8.8: Showing Staff Table content

Visitor:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO visitor(Vvisitor_id,Vemail,Vfirst_name,Vlast_n
ame)VALUES(1000,'averdin@miners.utep.edu','Alan','Verdin'),(1001,'gjones@miners.utep.edu','Garret','Jones'),(1002,'ssqua
repants@miners.utep.edu','Spongebob','Squarepants'),(1003,'msmith@miners.utep.edu','Morty','Smith');
```

Figure 8.9: Inserting values to Visitor Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from visitor;
+-------------+-----------------------------+-------------+-------------+
| Vvisitor_id | Vemail                      | Vfirst_name | Vlast_name  |
+-------------+-----------------------------+-------------+-------------+
|        1000 | averdin@miners.utep.edu     | Alan        | Verdin      |
|        1001 | gjones@miners.utep.edu      | Garret      | Jones       |
|        1002 | ssquarepants@miners.utep.edu | Spongebob  | Squarepants |
|        1003 | msmith@miners.utep.edu      | Morty       | Smith       |
+-------------+-----------------------------+-------------+-------------+
4 rows in set (0.0932 sec)
```

Figure 8.10: Showing Visitor Table content

Visitor_phone:

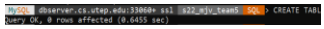
```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO visitor_phone(1000,'(915)123-9876'),(1000,'(91
5)123-4561'),(1001,'(915)456-1234'),(1001,'(915)789-001'),(1002,'(156)789-4578');
```

Figure 8.11: Inserting values to Visitor_phone Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from visitor_phone
                                                        -> ;
+-------------+---------------+
| Vvisitor_id | Vphone        |
+-------------+---------------+
|        1000 | (915)123-9876 |
|        1000 | (915)123-4561 |
|        1001 | (915)456-1234 |
|        1001 | (915)789-0001 |
|        1002 | (156)789-4578 |
+-------------+---------------+
5 rows in set (0.0688 sec)
```

Figure 8.12: Showing Visitor_phone Table content

Request:



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO request(Rrequest_id,Rrequest_type,Rvisitor_typ
e,Rstatus,Rdescription)VALUES(5001,'Email','Student','In progress','Need Advising'),
                                                    -> (5002,'Phone','Student','Pending','Need Hold Removal'),
                                                    -> (5003,'Phone','Faculty','In Progress','Payroll Issues),
                                                    -> (5004,'Walkin','Faculty','Pending','Payroll Issues),
                                                    -> (5005,'Phone','Faculty','Closed','Drop Student'),
                                                    -> (5006,'Email','Staff','Closed','Schedule Update'),
                                                    -> (5007,'Walkin','Staff','Pending','Need Tech Support'),
                                                    -> (5008,'Walkin','Student','In Progress','Registration Help'
),
                                                    -> (5009,'Phone','Student','Deleted','Need Advising'),
                                                    -> (5010,'Walkin','Faculty','Deleted','Drop Student');
```

Figure 8.13: Inserting values to Request Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from request;
+-------------+---------------+---------------+-------------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription      |
+-------------+---------------+---------------+-------------+-------------------+
|        5001 | Email         | Student       | In Progress | Need Advising     |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal |
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues    |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues    |
|        5005 | Phone         | Faculty       | Closed      | Drop Student      |
|        5006 | Email         | Staff         | Closed      | Schedule Update   |
|        5007 | Walkin        | Staff         | Pending     | Need Tech Support |
|        5008 | Walkin        | Student       | In Progress | Registration Help |
|        5009 | Phone         | Student       | Deleted     | Need Advsing      |
|        5010 | Walkin        | Faculty       | Deleted     | Drop Student      |
+-------------+---------------+---------------+-------------+-------------------+
10 rows in set (0.0408 sec)
```

Figure 8.14: Showing Request Table content

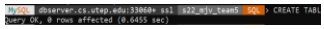
Request_destination:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO request_destination VALUES ("Need Advising","Advising Center"),
                                                    -> ("Need Hold Removal","Advising Center"),
                                                    -> ("Payroll Isuues","Employee Support"),
                                                    -> ("Drop Student","Academic Center"),
                                                    -> ("Schedule Update","Employee Support"),
                                                    -> ("Need Tech Support","Support Desk"),
                                                    -> ("Registration Help","Academic Center");
Query OK, 7 rows affected (0.3774 sec)

Records: 7  Duplicates: 0  Warnings: 0
```

Figure 8.15: Inserting Values into Request _destination Table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from request_destination;
+-------------------+------------------+
| Rdescription      | Rforward_to      |
+-------------------+------------------+
| Drop Student      | Academic Center  |
| Need Advising     | Advising Center  |
| Need Hold Removal | Advising Center  |
| Need Tech Support | Support Desk     |
| Payroll Isuues    | Employee Support |
| Registration Help | Academic Center  |
| Schedule Update   | Employee Support |
+-------------------+------------------+
7 rows in set (0.0251 sec)
```

Figure 8.16: Showing Request_destination Table content



Creates:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > INSERT INTO creates VALUES ("3/25/2022","10:00 AM MT","1000","5008"),
                                                          -> ("3/26/2022","11:15 AM MT","1002","5007"),
                                                          -> ("2/10/2022","7:30 AM MT","1003","5004"),
                                                          -> ("3/15/2022","3:00 PM MT","1000","5010");
Query OK, 4 rows affected (0.2748 sec)
```

Figure 8.17: Inserting values into creates table

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from creates;
+-----------+-------------+-------------+-------------+
| Cdate     | Ctime       | Vvisitor_ID | Rrequest_ID |
+-----------+-------------+-------------+-------------+
| 3/25/2022 | 10:00 AM MT |        1000 |        5008 |
| 3/26/2022 | 11:15 AM MT |        1002 |        5007 |
| 2/10/2022 | 7:30 AM MT  |        1003 |        5004 |
| 3/15/2022 | 3:00 PM MT  |        1000 |        5010 |
+-----------+-------------+-------------+-------------+
4 rows in set (0.0270 sec)
```

Figure 8.18: Showing Creates Table content

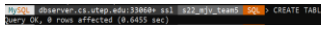
9. SQL QUERIES

How many visitors are assisted per day, week, semester or year, or any other period of time?

The following queries will allow assistant personnel to determine the number of requests per day, week (out of 52), month, year, and semester.

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select day(date_requested), count(*) as
 'Number of requests made per day' from request_date group by day(date_requested) order by day(date_r
equested);
+---------------------+---------------------------------+
| day(date_requested) | Number of requests made per day |
+---------------------+---------------------------------+
|                   2 |                               1 |
|                   5 |                               1 |
|                  10 |                               2 |
|                  12 |                               1 |
|                  15 |                               1 |
|                  25 |                               2 |
|                  26 |                               1 |
|                  28 |                               1 |
+---------------------+---------------------------------+
8 rows in set (0.0504 sec)
```

Figure 9.1: Showing number of requests made per day



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select year(date_requested), week(
date_requested), count(*) as 'Requests per week' from request_date group by week(date_requested)
 order by week(date_requested);
+----------------------+----------------------+--------------------+
| year(date_requested) | week(date_requested) | Requests per week |
+----------------------+----------------------+--------------------+
|                 2022 |                    6 |                  2 |
|                 2022 |                    9 |                  2 |
|                 2022 |                   10 |                  1 |
|                 2022 |                   11 |                  1 |
|                 2022 |                   12 |                  3 |
|                 2021 |                   34 |                  1 |
+----------------------+----------------------+--------------------+
6 rows in set (0.0564 sec)
```

Figure 9.2: Showing number of requests made per week

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select year(date_requested), month(date_requested),
count(*) as 'Requests per month' from request_date group by month(date_requested) order by month(date_requested);
+----------------------+----------------------+--------------------+
| year(date_requested) | month(date_requested) | Requests per month |
+----------------------+----------------------+--------------------+
|                 2022 |                    2 |                  2 |
|                 2022 |                    3 |                  7 |
|                 2021 |                    8 |                  1 |
+----------------------+----------------------+--------------------+
3 rows in set (0.0495 sec)
```

Figure 9.3: Showing number of requests made per month

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select year(date_requested), count(*)
as 'Requests per year' from request_date group by year(date_requested) order by year(date_requested)
;
+----------------------+--------------------+
| year(date_requested) | Requests per year |
+----------------------+--------------------+
|                 2021 |                  1 |
|                 2022 |                  9 |
+----------------------+--------------------+
2 rows in set (0.0602 sec)
```

Figure 9.4: Showing number of requests made per year

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select count(*) as 'Number of requests made in
spring semester' from request_date where date_requested between '2022-01-18' AND '2022-05-15';
+-----------------------------------------+
| Number of requests made in spring semester |
+-----------------------------------------+
|                                       9 |
+-----------------------------------------+
1 row in set (0.0494 sec)
```

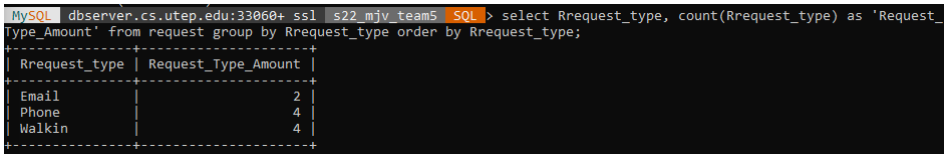
Figure 9.3: Showing number of requests made per semester

How many requests are done by email, telephone, or by walk-in?

This query counts the types of requests and pairs it with the request type in one table. It shows the exact number of requests done by each category, in this case 2 requests done by email, 4 requests done by telephone, and 4 requests done by walk-in. Here, we selected everything for Rrequest_type and used

the count function to count how many requests each category has, then we listed it as a table along Rrequest_type under the column name of Request_Type_Amount. This query will help us pull out the information we need regarding email,telephone or walk-in, for example being able to compare data between each category and determining which category is the most and least requested.

CMD: select Rrequest_type, count(Rrequest_type) as 'Request_Type_Amount' from request group by Rrequest_type order by Rrequest_type;

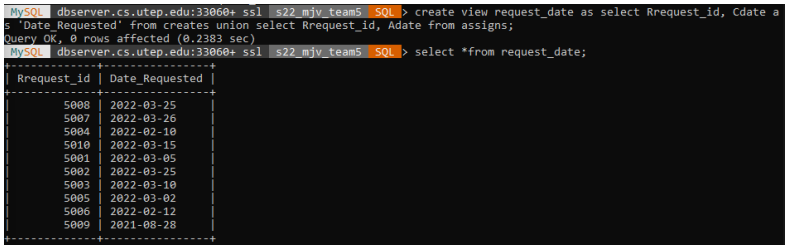


```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select Rrequest_type, count(Rrequest_type) as 'Request_
Type_Amount' from request group by Rrequest_type order by Rrequest_type;
+---------------+--------------------+
| Rrequest_type | Request_Type_Amount |
+---------------+--------------------+
| Email         |                  2 |
| Phone         |                  4 |
| Walkin        |                  4 |
+---------------+--------------------+
```

Figure 9.6: Showing number of requests done by email, phone, and Walkin.

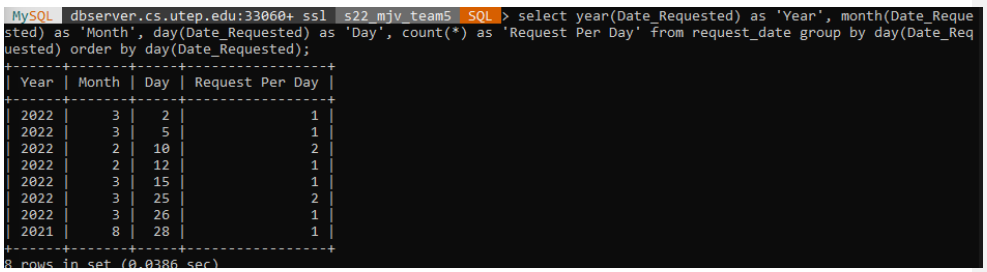
Days of the week with most visits?

For this query we went ahead and created a view table request_date that merges the requests from assigns and creates to one table. From there it was easier to query the requests per day.



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > create view request_date as select Rrequest_id, Cdate a
s 'Date_Requested' from creates union select Rrequest_id, Adate from assigns;
Query OK, 0 rows affected (0.2383 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select *from request_date;
+-------------+----------------+
| Rrequest_id | Date_Requested |
+-------------+----------------+
|        5008 | 2022-03-25     |
|        5007 | 2022-03-26     |
|        5004 | 2022-02-10     |
|        5010 | 2022-03-15     |
|        5001 | 2022-03-05     |
|        5002 | 2022-03-25     |
|        5003 | 2022-03-10     |
|        5005 | 2022-03-02     |
|        5006 | 2022-02-12     |
|        5009 | 2021-08-28     |
+-------------+----------------+
```

CMD: select year(Date_Requested) as 'Year', month(Date_Requested) as 'Month', day(Date_Requested) as 'Day', count(*) as 'Request Per Day' from request_date group by day(Date_Requested) order by day(Date_Requested);



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select year(Date_Requested) as 'Year', month(Date_Reque
sted) as 'Month', day(Date_Requested) as 'Day', count(*) as 'Request Per Day' from request_date group by day(Date_Req
uested) order by day(Date_Requested);
+------+-------+-----+-----------------+
| Year | Month | Day | Request Per Day |
+------+-------+-----+-----------------+
| 2022 |     3 |   2 |               1 |
| 2022 |     3 |   5 |               1 |
| 2022 |     2 |  10 |               2 |
| 2022 |     2 |  12 |               1 |
| 2022 |     3 |  15 |               1 |
| 2022 |     3 |  25 |               2 |
| 2022 |     3 |  26 |               1 |
| 2021 |     8 |  28 |               1 |
+------+-------+-----+-----------------+
8 rows in set (0.0386 sec)
```

What are the most common requests?

-The following Query will take care of providing the total number of requests that were made depending on its description. This means that we will report the most common request types depending on their type. How does this query benefit us? This will help us identify much faster the type of request that is most requested in the system, making the problem detection process much faster for the staff. Also, this query is within the requirements that the system must have according to the client.

CMD: SELECT Rdescription, count(Rdescription) AS 'Most_Common_Request' FROM request GROUP BY Rdescription ORDER BY 'Most_Common_Request' DESC;

```
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT Rdescription, count(Rdescription) AS 'Most_Commo
n_Request' FROM request GROUP BY Rdescription ORDER BY 'Most_Common_Request' DESC;
+-------------------+---------------------+
| Rdescription      | Most_Common_Request |
+-------------------+---------------------+
| Need Advising     |                   1 |
| Need Hold Removal |                   1 |
| Payroll Issues    |                   2 |
| Drop Student      |                   2 |
| Schedule Update   |                   1 |
| Need Tech Support |                   1 |
| Registration Help |                   1 |
| Need Advsing      |                   1 |
+-------------------+---------------------+
```

Figure 9.9: Query to get the most common request

What is the most common type of visitor?

In the following command I use the value 1 to place the most repeated value from the column Rvisitor_type alongside its count in a table. Since in our test data we have an equal number of visitors for students and faculty. I decided to use a 4 as the print value as it will show the four types of visitor types and count descending order.

CMD: SELECT Rvisitor_type, count(Rvisitor_type) AS 'Most_Common_Visitor' FROM request GROUP BY Rvisitor_type ORDER BY 'Most_Common_Visitor' DESC 1;

CMD: SELECT Rvisitor_type, count(Rvisitor_type) AS 'Most_Common_Visitor' FROM request GROUP BY Rvisitor_type ORDER BY 'Most_Common_Visitor' DESC 4;

```
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT Rvisitor_type, count(Rvisitor_type) AS 'Most_Com
mon_Visitor' FROM request GROUP BY Rvisitor_type ORDER BY 'Most_Common_Visitor' DESC LIMIT 1;
+---------------+---------------------+
| Rvisitor_type | Most_Common_Visitor |
+---------------+---------------------+
| Student       |                   4 |
+---------------+---------------------+
1 row in set (0.1660 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT Rvisitor_type, count(Rvisitor_type) AS 'Most_Com
mon_Visitor' FROM request GROUP BY Rvisitor_type ORDER BY 'Most_Common_Visitor' DESC LIMIT 2;
+---------------+---------------------+
| Rvisitor_type | Most_Common_Visitor |
+---------------+---------------------+
| Student       |                   4 |
| Faculty       |                   4 |
+---------------+---------------------+
2 rows in set (0.0916 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT Rvisitor_type, count(Rvisitor_type) AS 'Most_Com
mon_Visitor' FROM request GROUP BY Rvisitor_type ORDER BY 'Most_Common_Visitor' DESC LIMIT 4;
+---------------+---------------------+
| Rvisitor_type | Most_Common_Visitor |
+---------------+---------------------+
| Student       |                   4 |
| Faculty       |                   4 |
| Staff         |                   2 |
+---------------+---------------------+
3 rows in set (0.0545 sec)
```

How many requests are made by Email, Phone, or Walk-in?

-The following Queries will be in charge to provide specific information about the total amount of request were enter by category(Email,Phone,Walk-in). How do these queries help us? These queries will be very useful when we have to make a report with the most important types of information in our database. Within the requirements, we are asked to inform the total number of reports that were made depending on their category within the report. At the same time, this will help us to better identify what type of request is the most used by users who interact with the system.

CMD:

-For email:

SELECT COUNT(Rrequest_type) AS 'Requests done by Email' WHERE Rrequest_type='Email';

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT COUNT(Rrequest_type) as 'Requests d
one by Email' FROM request WHERE Rrequest_type='Email';
+------------------------+
| Requests done by Email |
+------------------------+
|                      2 |
+------------------------+
1 row in set (0.0600 sec)
```

Figure 9.11: Query to get the number of Request done by email

-For phone:

SELECT COUNT(Rrequest_type) AS 'Requests done by Email' WHERE Rrequest_type='Phone';

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT COUNT(Rrequest_type) as 'Requests d
one by Phone' FROM request WHERE Rrequest_type='Phone';
+------------------------+
| Requests done by Phone |
+------------------------+
|                      4 |
+------------------------+
1 row in set (0.0513 sec)
```

Figure 9.12: Query to get the number of Request done by Phone

-For Walk-in:

SELECT COUNT(Rrequest_type) AS 'Requests done by Email' WHERE Rrequest_type='Walkin';

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT COUNT(Rrequest_type) as 'Requests d
one by Walkin' FROM request WHERE Rrequest_type='Walkin';
+-------------------------+
| Requests done by Walkin |
+-------------------------+
|                       4 |
+-------------------------+
1 row in set (0.0496 sec)
```

Figure 9.13: Query to get the number of Request done by Walkin

10. GRAPHICAL USER INTERFACE

Alan Verdin

-The following images correspond to the index file that belongs only to team 5, that has the folder repository for each group member.

```
<div class="container">
    <h1>Select Subdirectory: </h1>

    <a href="DB4342_PhP_averdin/index.php">averdin</a><br>
    <a href="DB4342_PhP_cagomez15/index.php">cagomez15</a><br>
    <a href="DB4342_PhP_gwjones/index.php">gwjones</a><br>
    <a href="DB4342_PhP_maroberto6/index.php">maroberto6</a><br>
```

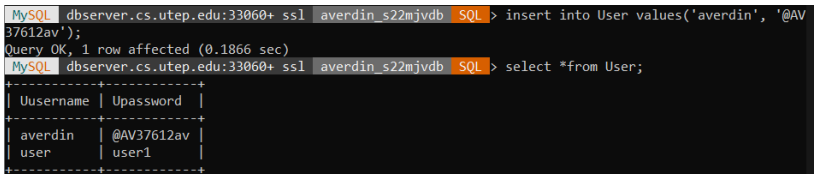
-The next image corresponds to my config file that has all of my user, server and host information.

```
$host = "dbserver.cs.utep.edu"; #enter the DB server location
$db = "averdin_s22mjvdb";   # 1. Enter your team database here for your group project.
          # OR 2. Enter your individual database here to complete this exercise.

$username = "averdin";  # If 1 above (for your group project), enter the username of the interface or reports lead.
                # If 2 above (for this individual exercise), enter your username.

$password = "@AV37612av";  # If 1 above (for your group project), enter the password of the interface or reports lea
                # If 2 above (for this individual exercise), enter your individual password.
```

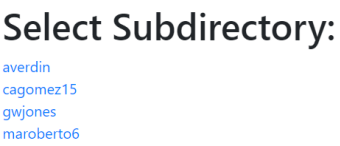
-The next images show the Student and User table created and inserted in MySQL to interact with the GUI when logging in and adding users.

```
MySQL dbserver.cs.utep.edu:33060+ ssl averdin_s22mjvdb SQL > CREATE TABLE Student(Sid VARCHAR(3) PRI
MARY KEY, SfirstName VARCHAR(30), SmiddleName VARCHAR(30), SlastName VARCHAR(30));
Query OK, 0 rows affected (0.7048 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl averdin_s22mjvdb SQL > CREATE TABLE User(Uusername VARCHAR(30)
 PRIMARY KEY, Upassword VARCHAR(30));
Query OK, 0 rows affected (0.4536 sec)
```

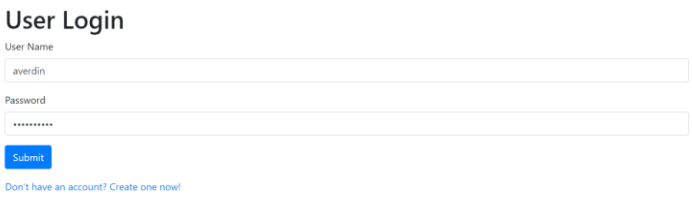
```
MySQL dbserver.cs.utep.edu:33060+ ssl averdin_s22mjvdb SQL > insert into Student values(1, 'Alan', '
', 'Verdin');
Query OK, 1 row affected (0.1909 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl averdin_s22mjvdb SQL > select *from Student;
+-----+------------+-------------+-----------+
| Sid | SfirstName | SmiddleName | SlastName |
+-----+------------+-------------+-----------+
| 1   | Alan       |             | Verdin    |
+-----+------------+-------------+-----------+
1 row in set (0.0050 sec)
```

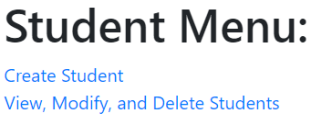


-I then added my repository to the team 5 repository and ran the server link to see if my GUI works. The image below shows the repositories of each member of the team.



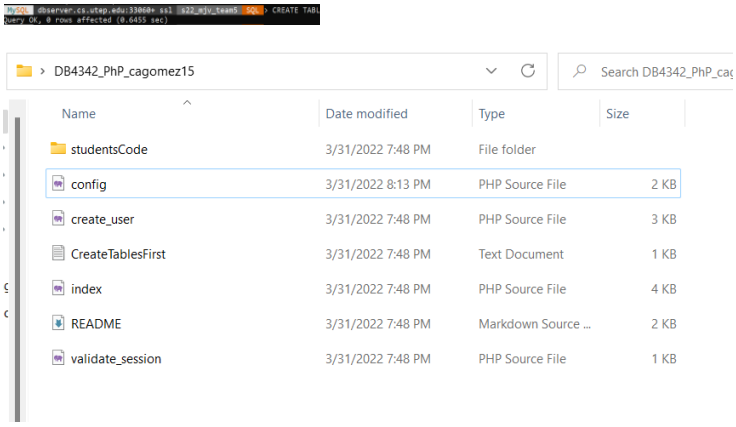
-The next images show the process of logging in to my GUI, using my credentials.





Christian Gomez

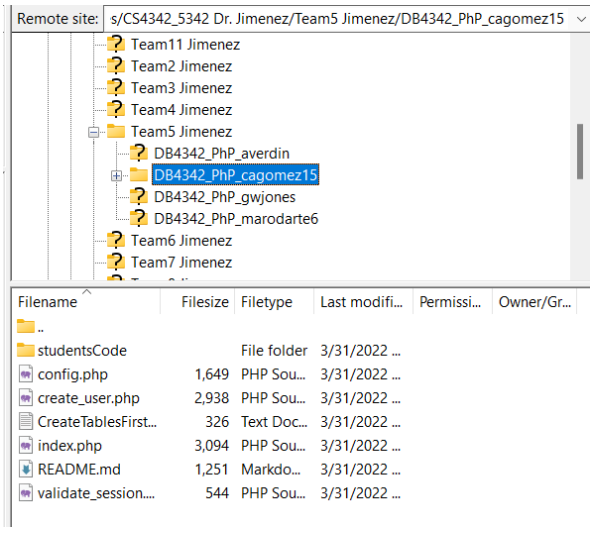
-File Document with my username:



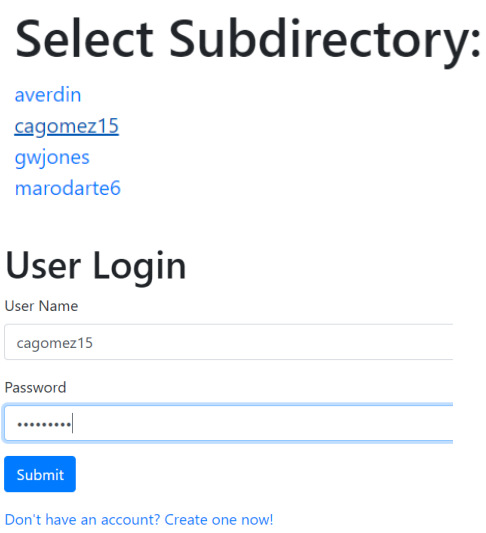
-My config modification:

```php
<?php

$host = "dbserver.cs.utep.edu"; #enter the DB server location
$db = "cagomez15_s22mjvdb";   # 1. Enter your team database here for your group project.
        # OR 2. Enter your individual database here to complete this exercise.

$username = "cagomez15";  # If 1 above (for your group project), enter the username of th
            # If 2 above (for this individual exercise), enter your username.

$password = "1998ABjk!";  # If 1 above (for your group project), enter the password of th
            # If 2 above (for this individual exercise), enter your individual passw
```

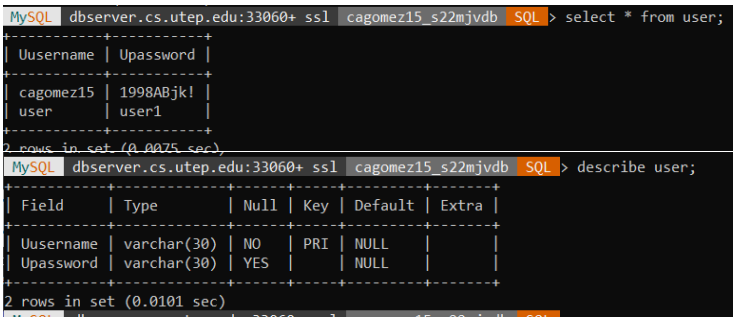
-My FileZilla menu:

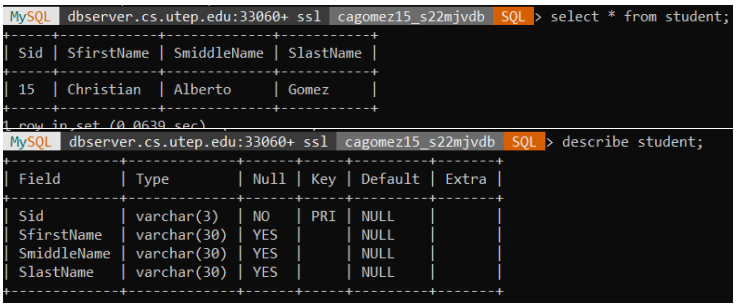


-Log In:



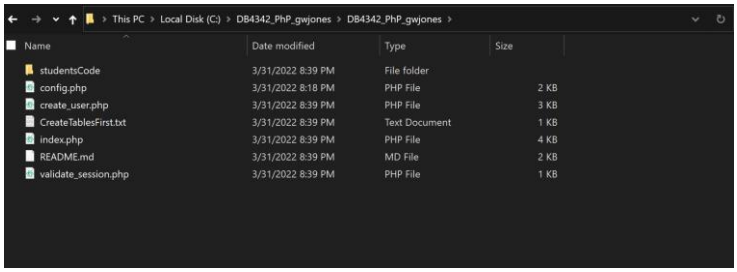
-User and Student tables from my personal database:



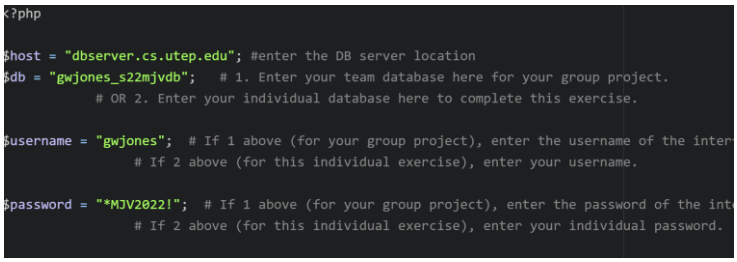


Garret Jones

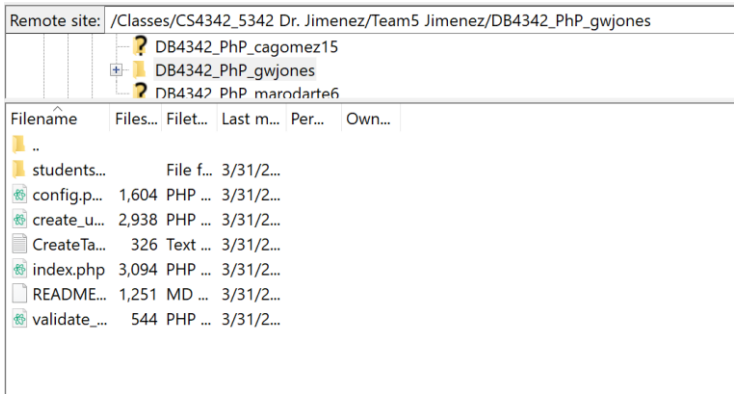
-file doc with username:



-config file:



FileZilla menu:



Login:

Select Subdirectory:

averdin
cagomez15
gwjones
marodarte6

User Login

User Name

gwjones

Password

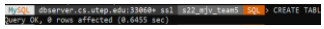
••••••••

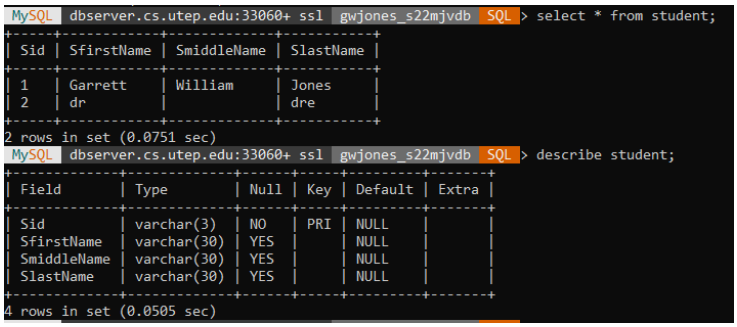
Submit

Don't have an account? Create one now!

Individual User and Student tables:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  gwjones_s22mjvdb  SQL > select * from user;
+-----------+-----------+
| Uusername | Upassword |
+-----------+-----------+
| gwjones   | *MJV2022! |
+-----------+-----------+
1 row in set (0.0522 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  gwjones_s22mjvdb  SQL > describe user;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Uusername | varchar(30) | NO   | PRI | NULL    |       |
| Upassword | varchar(30) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.0480 sec)
```

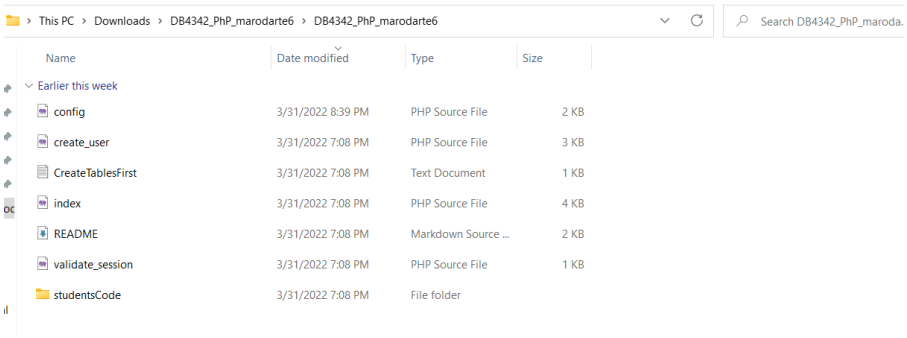




```
MySQL  dbserver.cs.utep.edu:33060+ ssl  gwjones_s22mjvdb  SQL > select * from student;
+-----+------------+-------------+-----------+
| Sid | SfirstName | SmiddleName | SlastName |
+-----+------------+-------------+-----------+
| 1   | Garrett    | William     | Jones     |
| 2   | dr         |             | dre       |
+-----+------------+-------------+-----------+
2 rows in set (0.0751 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  gwjones_s22mjvdb  SQL > describe student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Sid         | varchar(3)  | NO   | PRI | NULL    |       |
| SfirstName  | varchar(30) | YES  |     | NULL    |       |
| SmiddleName | varchar(30) | YES  |     | NULL    |       |
| SlastName   | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.0505 sec)
```

Miguel Rodarte

-File document with my username:

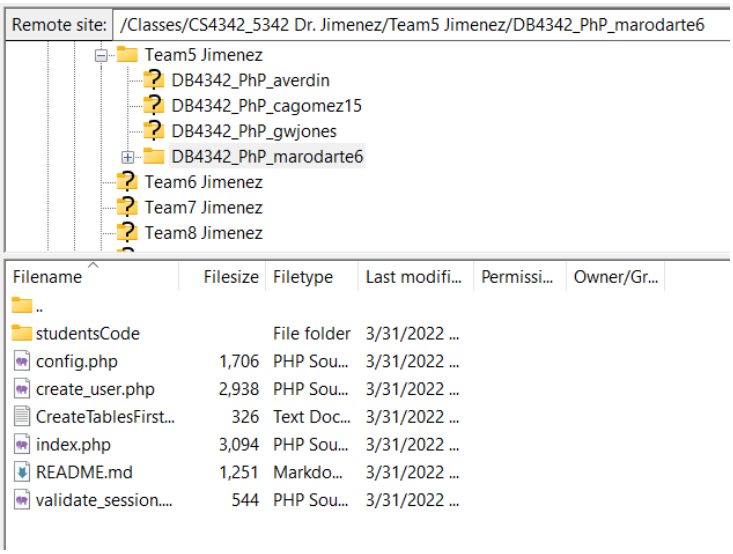


-Config file:

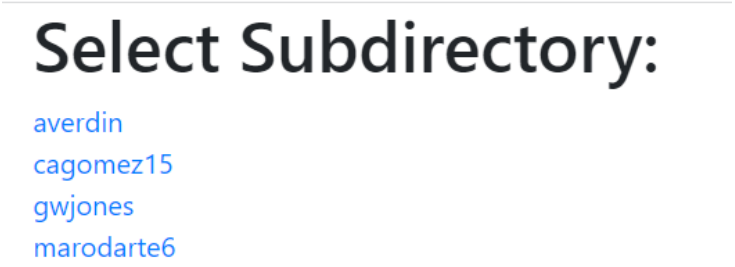
```php
<?php

$host = "dbserver.cs.utep.edu"; #enter the DB server location
$db = "marodarte6_s22mjvdb";   # 1. Enter your team database here for your group project.
        # OR 2. Enter your individual database here to complete this exercise.

$username = "marodarte6";  # If 1 above (for your group project), enter the username of the interface or reports lead.
            # If 2 above (for this individual exercise), enter your username.

$password = "mrodBCBS1!";  # If 1 above (for your group project), enter the password of the interface or reports lead.
            # If 2 above (for this individual exercise), enter your individual password.
```

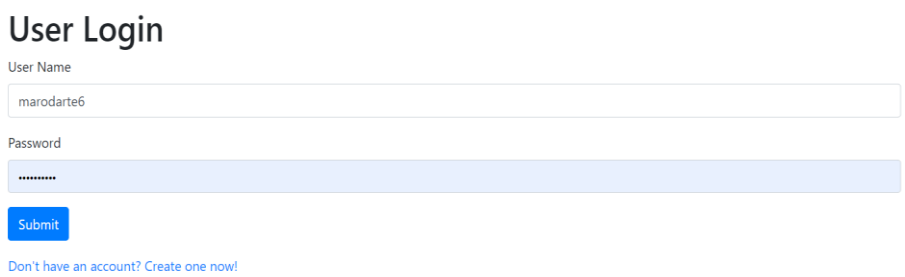
-FileZilla Menu:


CREATE TABL
Query OK, 0 rows affected (0.6455 sec)


Remote site: /Classes/CS4342_5342 Dr. Jimenez/Team5 Jimenez/DB4342_PhP_marodarte6
Team5 Jimenez
DB4342_PhP_averdin
DB4342_PhP_cagomez15
DB4342_PhP_gwjones
DB4342_PhP_marodarte6
Team6 Jimenez
Team7 Jimenez
Team8 Jimenez
Filename
Filesize
Filetype
Last modifi...
Permissi...
Owner/Gr...
..
studentsCode
File folder
3/31/2022 ...
config.php
1,706
PHP Sou...
3/31/2022 ...
create_user.php
2,938
PHP Sou...
3/31/2022 ...
CreateTablesFirst...
326
Text Doc...
3/31/2022 ...
index.php
3,094
PHP Sou...
3/31/2022 ...
README.md
1,251
Markdo...
3/31/2022 ...
validate_session....
544
PHP Sou...
3/31/2022 ...

-Login:


Select Subdirectory:
averdin
cagomez15
gwjones
marodarte6



-User and Student tables from personal database:

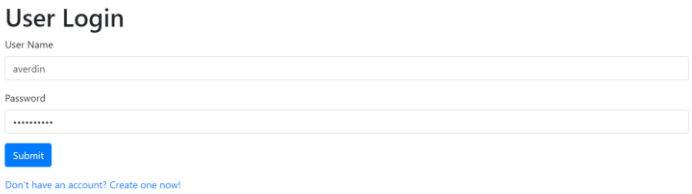
```
MySQL  dbserver.cs.utep.edu:33060+ ssl  marodarte6_s22mjvdb  SQL > select*from user;
+------------+------------+
| Uusername  | Upassword  |
+------------+------------+
| marodarte6 | mrodBCBS1! |
| user       | user1      |
+------------+------------+
2 rows in set (0.0516 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  marodarte6_s22mjvdb  SQL > describe user;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Uusername | varchar(30) | NO   | PRI | NULL    |       |
| Upassword | varchar(30) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.0616 sec)
```

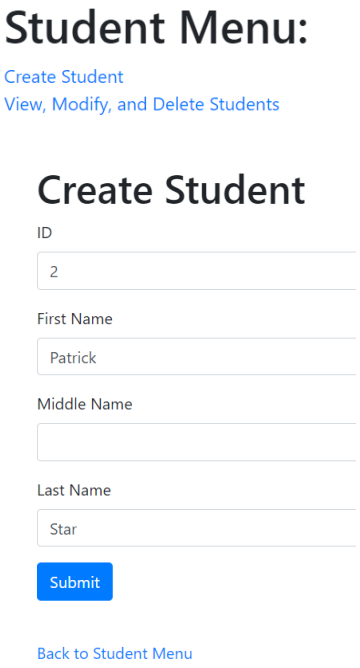
```
MySQL  dbserver.cs.utep.edu:33060+ ssl  marodarte6_s22mjvdb  SQL > select*from student;
+-----+------------+-------------+-----------+
| Sid | SfirstName | SmiddleName | SlastName |
+-----+------------+-------------+-----------+
| 1   | Miguel     | Angel       | Rodarte   |
+-----+------------+-------------+-----------+
1 row in set (0.0880 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  marodarte6_s22mjvdb  SQL > describe student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Sid         | varchar(3)  | NO   | PRI | NULL    |       |
| SfirstName  | varchar(30) | YES  |     | NULL    |       |
| SmiddleName | varchar(30) | YES  |     | NULL    |       |
| SlastName   | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.0786 sec)
```

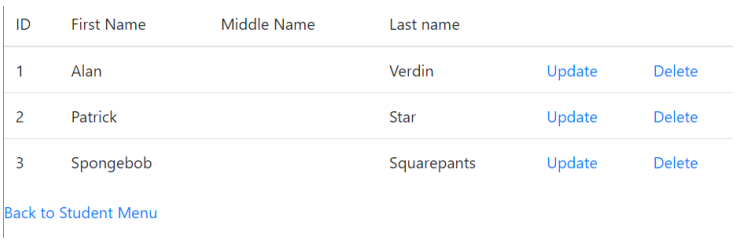
11. ISS WEB SERVER AND GUI

Alan Verdin

The images below show how I created a new user and logged on to the server with the GUI.

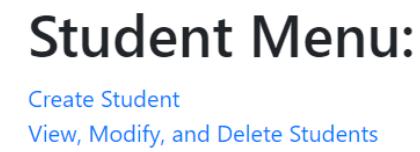






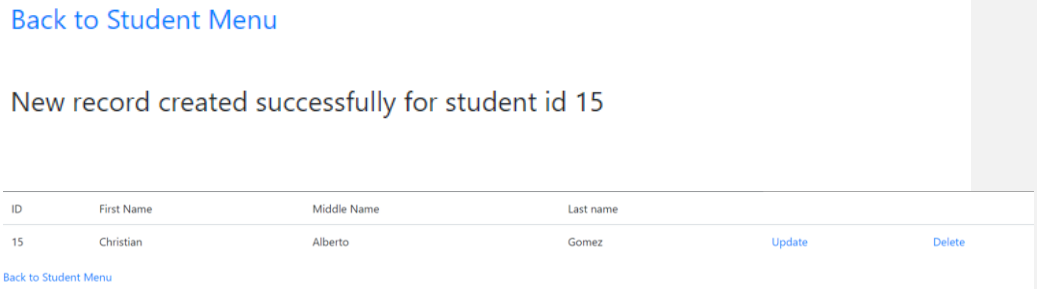
Christian Gomez

-Here is the menu after logging in:



-Create Student from my point of view:

-result:



Garrett Jones

-Menu

Student Menu:

Create Student
View, Modify, and Delete Students

-create student:

Create Student

ID

5

First Name

John

Middle Name

Last Name

Smith

Submit

-result:

ID	First Name	Middle Name	Last name		
1	Garrett	William	Jones	Update	Delete
2	dr		dre	Update	Delete
5	John		Smith	Update	Delete

Back to Student Menu

Miguel Rodarte

-Menu after logging in:

Student Menu:

Create Student

View, Modify, and Delete Students

-Create Student:

Create Student

ID

2

First Name

Jane

Middle Name

Last Name

Doe

Submit

Back to Student Menu

-Result:

ID	First Name	Middle Name	Last name		
1	Miguel	Angel	Rodarte	Update	Delete
2	Jane		Doe	Update	Delete

Back to Student Menu

13. VIEWS

-Christian Gomez:

This View will provide information about the most common type of request. In other words, the view will print the frequency of each request.

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE VIEW Most_Common_Request
                                                          -> AS
                                                          -> SELECT Rdescription,COUNT(*)
                                                          -> FROM Request
                                                          -> GROUP BY Rdescription;
Query OK, 0 rows affected (0.1734 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from Most_Common_Request;
+-------------------+----------+
| Rdescription      | COUNT(*) |
+-------------------+----------+
| Need Advising     |        1 |
| Need Hold Removal |        1 |
| Payroll Issues    |        2 |
| Drop Student      |        2 |
| Schedule Update   |        1 |
| Need Tech Support |        1 |
| Registration Help |        1 |
| Need Advsing      |        1 |
+-------------------+----------+
8 rows in set (0.0782 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL >
```

Figure 13.1: Creation of Most_Common_Request VIEW

-Garrett Jones:

This view provides information about how many visitors assisted per day, week, year, semester

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE VIEW Visitors_Per_d
ay AS SELECT day(date_requested), count(*) as 'Number of requests made per day' FROM req
uest_date GROUP BY day(date_requested) ORDER BY day(date_requested);
Query OK, 0 rows affected (0.1330 sec)
```

Figure 13.2: Creation of Requests_per_day View

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT * FROM Visitors_per
_day;
+---------------------+--------------------------------+
| day(date_requested) | Number of requests made per day |
+---------------------+--------------------------------+
|                   2 |                              1 |
|                   5 |                              1 |
|                  10 |                              2 |
|                  12 |                              1 |
|                  15 |                              1 |
|                  25 |                              2 |
|                  26 |                              1 |
|                  28 |                              1 |
+---------------------+--------------------------------+
8 rows in set (0.0414 sec)
```

Figure 13.3: SELECT * FROM Visitors_per_day

```
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > CREATE VIEW Requests_per_w
eek AS SELECT year(date_requested), week(date_requested), count(*) as 'Requests per week
' from request_date group by week(date_requested) ORDER BY week(date_requested);
Query OK, 0 rows affected (0.1207 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT * FROM requests_per
_week
                                                       -> ;
+----------------------+----------------------+-------------------+
| year(date_requested) | week(date_requested) | Requests per week |
+----------------------+----------------------+-------------------+
|                 2022 |                    6 |                 2 |
|                 2022 |                    9 |                 2 |
|                 2022 |                   10 |                 1 |
|                 2022 |                   11 |                 1 |
|                 2022 |                   12 |                 3 |
|                 2021 |                   34 |                 1 |
+----------------------+----------------------+-------------------+
6 rows in set (0.0045 sec)
```

Figure 13.4: Creation of Requests_per_week View

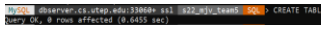
```
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > CREATE VIEW Requests_per_y
ear AS SELECT year(date_requested), count(*) AS 'Requests_per_year' FROM request_date GR
OUP BY year(date_requested) ORDER BY year(date_requested);
Query OK, 0 rows affected (0.1656 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT * FROM Requests_per
_year;
+----------------------+-------------------+
| year(date_requested) | Requests_per_year |
+----------------------+-------------------+
|                 2021 |                 1 |
|                 2022 |                 9 |
+----------------------+-------------------+
2 rows in set (0.0542 sec)
```

Figure 13.5: Creation of Requests_per_year View and SELECT Statement

```
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > CREATE VIEW Requests_per_s
emester AS SELECT count(*) AS 'Number of requests made in Spring semester' FROM request_
date WHERE date_requested between '2022-01-18' AND '2022-05-15';
Query OK, 0 rows affected (0.1878 sec)
MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team5 SQL > SELECT * FROM Requests_per
_semester;
+-------------------------------------------+
| Number of requests made in Spring semester |
+-------------------------------------------+
|                                         9 |
+-------------------------------------------+
1 row in set (0.0067 sec)
```

Figure 13.6: Creation of Requests_per_semester view and SELECT statement

-Alan: Most_Common_Visitor view, shows all visitor types by count.



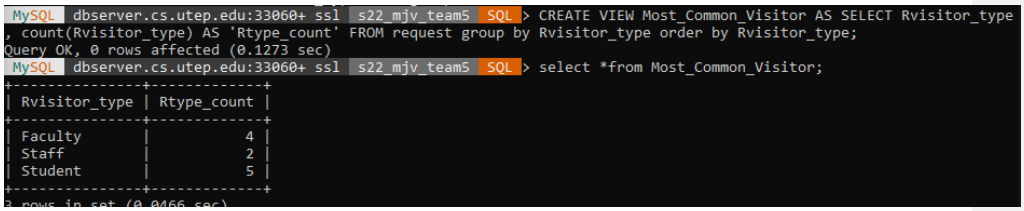


Figure 13.7: Creation of Most_Common_Visitor

-Miguel Rodarte: VIEW created for how many requests are done by email, phone, or walk-in.

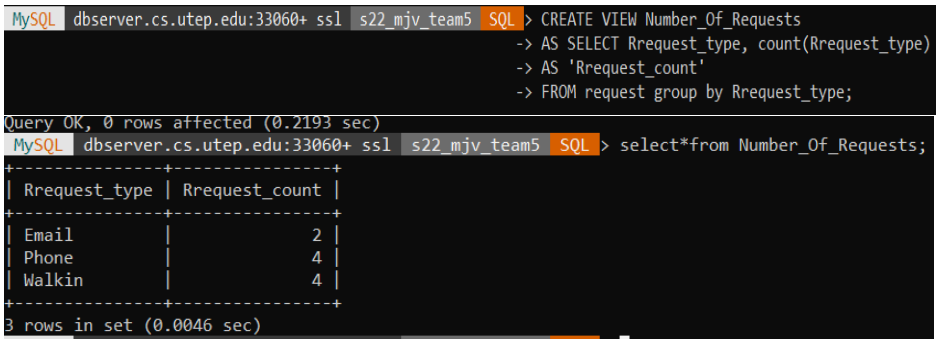


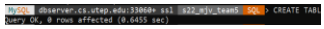
Figure 13.8: Creation of view for most common request type

14. Procedures

-Christian Gomez:

1) Procedure addRequestByVisitor.

-This procedure will be in charge of adding a request by the visitor. Once the visitor inputs all the required data then the procedure will be in charge of adding information to Request and Creates tables.



```
s22_mjv_team5  SQL > DELIMITER &
s22_mjv_team5  SQL > CREATE PROCEDURE addRequestByVisitor(
                  -> in visitorID INT,
                  -> in date varchar(100),
                  -> in time varchar(100),
                  -> in rId INT,
                  -> in firstName varchar(100),
                  -> in lastName varchar(100),
                  -> in email varchar(100),
                  -> in rType varchar(100),
                  -> in vType varchar(100),
                  -> in status varchar(100),
                  -> in description varchar(100))
                  -> BEGIN
                  -> INSERT INTO Request(Rrequest_id,Rrequest_type,Rvisitor_type,Rstatus,Rdescription)
                  -> VALUES(rId,rType,vType,status,description);
                  -> INSERT INTO creates(Cdate,Ctime,Vvisitor_ID,Rrequest_id)
                  -> VALUES(date,time,visitorID,rId);
                  -> INSERT INTO Visitor(Vvisitor_id,Vemail,Vfirst_name,Vlast_name)
                  -> VALUES(visitorID,email,firstName,lastName);
                  -> END&
```

Figure 14.1: Creation of procedure addRequestByVisitor – By Christian Gomez

2) **procedure addRequestByStaff.**

- -This procedure will be in charge of adding a request by the staff member. Once the visitor inputs all the required data then the procedure will be in charge of adding information to Request and Creates tables.

```
s22_mjv_team5  SQL > DELIMITER &
s22_mjv_team5  SQL > CREATE PROCEDURE addRequestByStaff(
                  -> in visitorID INT,
                  -> in date varchar(100),
                  -> in time varchar(100),
                  -> in rId INT,
                  -> in firstName varchar(100),
                  -> in lastName varchar(100),
                  -> in email varchar(100),
                  -> in rType varchar(100),
                  -> in vType varchar(100),
                  -> in status varchar(100),
                  -> in description varchar(100))
                  -> BEGIN
                  -> INSERT INTO Request(Rrequest_id,Rrequest_type,Rvisitor_type,Rstatus,Rdescription)
                  -> VALUES(rId,rType,vType,status,description);
                  -> INSERT INTO creates(Cdate,Ctime,Vvisitor_ID,Rrequest_id)
                  -> VALUES(date,time,visitorID,rId);
                  -> INSERT INTO Visitor(Vvisitor_id,Vemail,Vfirst_name,Vlast_name)
                  -> VALUES(visitorID,email,firstName,lastName);
                  -> END&
```

Figure 14.2: Creation of procedure addRequestByStaff – By Christian Gomez

3) **procedure assingsByStaff.**

-This procedure will be in charge of adding a information to table assigns. Information such as Date, Time, Staff ID and Request ID will be added to this table.

```
s22_mjv_team5  SQL > DELIMITER &
s22_mjv_team5  SQL > CREATE PROCEDURE assignsByStaff(
                  -> in dateIn varchar(100),
                  -> in timeIn varchar(100),
                  -> in staffID int,
                  -> in requestID int)
                  -> BEGIN
                  -> INSERT INTO assigns VALUES(dateIn,timeIn,staffID,requestID);
                  -> END&
```

Figure 14.3: Creation of procedure assignsByStaff – By Christian Gomez

-Alan: Procedure getVisitorType(visitor_type) returns a selection of only the wanted visitor type.

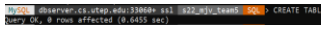
```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > DELIMITER &
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE PROCEDURE getVisitorType( in visitor_type varcha
r(100)) BEGIN SELECT *FROM request WHERE Rvisitor_type = visitor_type; END&
Query OK, 0 rows affected (0.2303 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > DELIMITER ;
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > call getVisitorType('Staff');
+-------------+---------------+---------------+---------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus | Rdescription      |
+-------------+---------------+---------------+---------+-------------------+
|        5006 | Email         | Staff         | Closed  | Schedule Update   |
|        5007 | Walkin        | Staff         | Pending | Need Tech Support |
+-------------+---------------+---------------+---------+-------------------+
2 rows in set (0.0484 sec)

Query OK, 0 rows affected (0.0484 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > call getVisitorType('Faculty');
+-------------+---------------+---------------+-------------+----------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription   |
+-------------+---------------+---------------+-------------+----------------+
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues |
|        5005 | Phone         | Faculty       | Closed      | Drop Student   |
|        5010 | Walkin        | Faculty       | Deleted     | Drop Student   |
+-------------+---------------+---------------+-------------+----------------+
4 rows in set (0.0474 sec)

Query OK, 0 rows affected (0.0474 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > call getVisitorType('Student');
+-------------+---------------+---------------+-------------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription      |
+-------------+---------------+---------------+-------------+-------------------+
|        1998 | Phone         | Student       | Pending     | Payroll Issues    |
|        5001 | Email         | Student       | In Progress | Need Advising     |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal |
|        5008 | Walkin        | Student       | In Progress | Registration Help |
|        5009 | Phone         | Student       | Deleted     | Need Advsing      |
+-------------+---------------+---------------+-------------+-------------------+
5 rows in set (0.0466 sec)
```

Figure 14.4: Creation of procedure getVisitorType – By Alan Verdin

-Garrett Jones: DeleteByRequest_ID.

Arguments: Request_ID_In – request ID to be deleted. Staff_ID_In- the staff member deleting the request.

Description: Updates the status of the request in the request table and inserts a new record with the date and time, staff ID of the staff member who deletes the request, as well as the request ID into deletes.

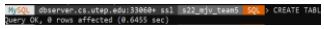


```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from request;
+-------------+---------------+---------------+-------------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription      |
+-------------+---------------+---------------+-------------+-------------------+
|        5001 | Email         | Student       | In Progress | Need Advising     |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal |
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues    |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues    |
|        5005 | Phone         | Faculty       | Closed      | Drop Student      |
|        5006 | Email         | Staff         | Closed      | Schedule Update   |
|        5007 | Walkin        | Staff         | Pending     | Need Tech Support |
|        5008 | Walkin        | Student       | In Progress | Need Advising     |
|        5009 | Phone         | Student       | Deleted     | Need Advsing      |
|        5010 | Walkin        | Faculty       | Deleted     | Drop Student      |
+-------------+---------------+---------------+-------------+-------------------+
10 rows in set (0.0429 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT * FROM deletes;
+------------+------------+-----------+-------------+
| Ddate      | Dtime      | Sstaff_ID | Rrequest_id |
+------------+------------+-----------+-------------+
| 2021-08-22 | 3:25 PM MT |      4001 |        5010 |
| 2021-09-05 | 1:05 PM MT |      4001 |        5009 |
| 2022-04-28 | 23:25:28   |      4003 |         327 |
| 2022-04-29 | 00:00:05   |      4003 |        5009 |
| 2022-04-29 | 00:00:23   |      4003 |        5009 |
| 2022-04-29 | 00:00:39   |      4003 |        5009 |
| 2022-04-29 | 00:05:26   |      4003 |         327 |
| 2022-04-29 | 00:18:55   |      4003 |         327 |
| 2022-04-29 | 00:19:22   |      4003 |        1612 |
+------------+------------+-----------+-------------+
9 rows in set (0.0493 sec)
```

Figure 14.5: Request and Deletes Tables Before Deletion – By Garrett Jones

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > DELIMITER $
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE PROCEDURE DeleteByRequest_ID(IN Request_ID_In INT,
 IN Staff_ID_In INT) BEGIN INSERT INTO deletes values(CURDATE(), CURTIME(), Staff_ID_In, Request_ID_In); SET FOREIGN_KE
Y_CHECKS=0; DELETE FROM request WHERE Rrequest_ID = Request_ID_In; SET FOREIGN_KEY_CHECKS=1; END $
Query OK, 0 rows affected (0.1977 sec)
```

Figure 14.6: Creation of procedure DeleteByRequest_ID – By Garrett Jones



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CALL DeleteByRequest_ID(5010, 4003);
Query OK, 0 rows affected (0.2502 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from request;
+-------------+--------------+---------------+-------------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription      |
+-------------+--------------+---------------+-------------+-------------------+
|        5001 | Email         | Student       | In Progress | Need Advising     |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal |
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues    |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues    |
|        5005 | Phone         | Faculty       | Closed      | Drop Student      |
|        5006 | Email         | Staff         | Closed      | Schedule Update   |
|        5007 | Walkin        | Staff         | Pending     | Need Tech Support |
|        5008 | Walkin        | Student       | In Progress | Need Advising     |
|        5009 | Phone         | Student       | Deleted     | Need Advsing      |
+-------------+--------------+---------------+-------------+-------------------+
9 rows in set (0.0542 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > SELECT * FROM deletes;
+------------+------------+-----------+-------------+
| Ddate      | Dtime      | Sstaff_ID | Rrequest_id |
+------------+------------+-----------+-------------+
| 2021-08-22 | 3:25 PM MT |      4001 |        5010 |
| 2021-09-05 | 1:05 PM MT |      4001 |        5009 |
| 2022-04-28 | 23:25:28   |      4003 |         327 |
| 2022-04-29 | 00:00:05   |      4003 |        5009 |
| 2022-04-29 | 00:00:23   |      4003 |        5009 |
| 2022-04-29 | 00:00:39   |      4003 |        5009 |
| 2022-04-29 | 00:05:26   |      4003 |         327 |
| 2022-04-29 | 00:18:55   |      4003 |         327 |
| 2022-04-29 | 00:19:22   |      4003 |        1612 |
| 2022-04-29 | 00:24:00   |      4003 |        5010 |
+------------+------------+-----------+-------------+
10 rows in set (0.0479 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL >
```

Figure 14.7: Request and Deletes Tables After Calling DeleteByRequest_ID – By Garrett Jones

-Miguel Rodarte:

This Procedure will be updating the request_type, visitor_type, status and/or description of a request.

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > DELIMITER $
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CREATE PROCEDURE UpdateRequest(
                                                        ->
                                                        -> IN staff_id int,
                                                        -> IN request_id int,
                                                        -> IN update_request_type VARCHAR(50),
                                                        -> IN update_visitor_type VARCHAR(50),
                                                        -> IN update_status VARCHAR(50),
                                                        -> IN update_description VARCHAR(100))
                                                        -> BEGIN
                                                        -> UPDATE request
                                                        -> SET
                                                        -> Rrequest_type = IFNULL(update_request_type,Rrequest_type),
                                                        -> Rvisitor_type = IFNULL(update_visitor_type,Rvisitor_type),
                                                        -> Rstatus = IFNULL(update_status,Rstatus),
                                                        -> Rdescription = IFNULL(update_description,Rdescription)
                                                        -> WHERE Rrequest_id = request_id;
                                                        -> INSERT INTO updates VALUES(CURDATE(), CURTIME(), staff_id, request_id);
                                                        -> END $
Query OK, 0 rows affected (0.2237 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > DELIMITER ;
```

Figure 14.8: Creation of procedure to update request.

Rrequest_id# 5008 BEFORE:

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from request;
+-------------+---------------+---------------+-------------+-------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription      |
+-------------+---------------+---------------+-------------+-------------------+
|        5001 | Email         | Student       | In Progress | Need Advising     |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal |
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues    |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues    |
|        5005 | Phone         | Faculty       | Closed      | Drop Student      |
|        5006 | Email         | Staff         | Closed      | Schedule Update   |
|        5007 | Walkin        | Staff         | Pending     | Need Tech Support |
|        5008 | Walkin        | Student       | In Progress | Need Advising     |
|        5009 | Phone         | Student       | Deleted     | Need Advsing      |
|        5010 | Walkin        | Faculty       | Deleted     | Drop Student      |
+-------------+---------------+---------------+-------------+-------------------+
10 rows in set (0.0042 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from updates;
+------------+-------------+-----------+-------------+
| Udate      | Utime       | Sstaff_ID | Rrequest_id |
+------------+-------------+-----------+-------------+
| 2022-03-05 | 11:00 AM MT |      4001 |        5001 |
| 2022-03-10 | 12:20 PM MT |      4002 |        5003 |
| 2022-03-02 | 8:00 AM MT  |      4003 |        5005 |
| 2022-02-12 | 6:29 PM MT  |      4003 |        5006 |
| 2022-03-25 | 10:20 AM MT |      4002 |        5008 |
| 2022-04-21 | 16:49:24    |      4001 |        5008 |
+------------+-------------+-----------+-------------+
6 rows in set (0.0039 sec)
```

Figure 14.9: Showing request table and updates table before.

Procedure Call: In this case, we will only be updating the request_type and status fields.

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > CALL UpdateRequest(4001,5008,'Email',NULL,'Pending',NULL);
Query OK, 1 row affected (0.5157 sec)
```

Figure 14.10: Calling Procedure to update two fields.

Rrequest_id# 5008 AFTER:

```
| 5007 | Walkin | Staff   | Pending | Need Tech Support |
| 5008 | Email  | Student | Pending | Need Advising     |
| 5009 | Phone  | Student | Deleted | Need Advsing      |
```

Figure 14.11: Request after being updated.

15. REPORTS

-Christian Gomez:

My contribution to the Report's Menu was the creation of Most Common Request Description and Most Common Visitor.

1. Most Common Request Description:

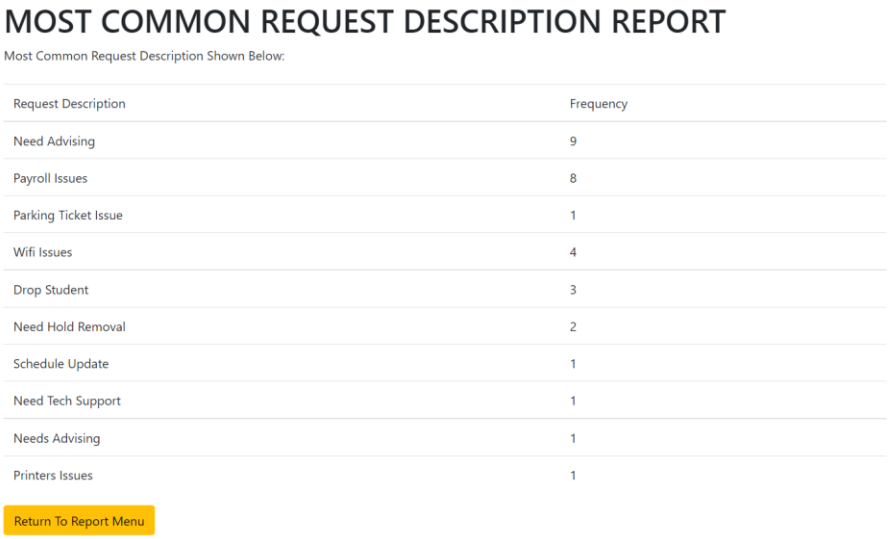


Figure 15.1: Interface for Most Common Request Description View

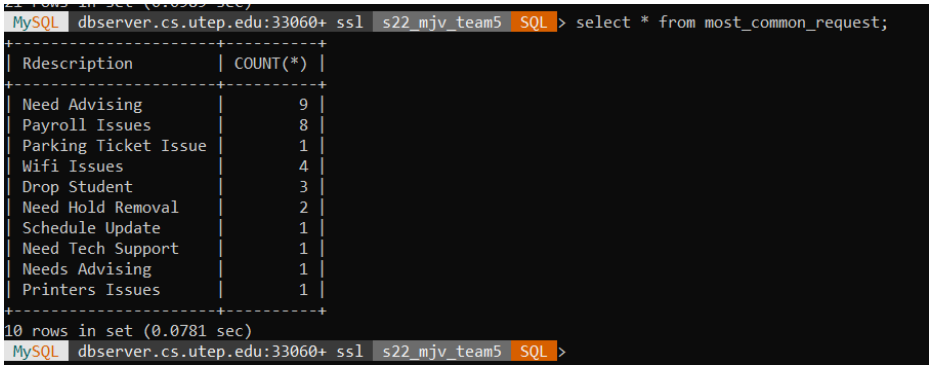


Figure 15.2: MySQL Most Common Request Description View

2. Most Common Visitor:

MOST COMMON VISTOR REPORT

Most Common Vistors Shown Below:

Visitor Type	Frequency
Faculty	7
Guest	3
Staff	13
Student	7
Walkin	1

Return To Report Menu

Figure 15.3: Interface for Most Common Visitor View

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select * from most_common_visitor;
+---------------+-------------+
| Rvisitor_type | Rtype_count |
+---------------+-------------+
| Faculty       |           7 |
| Guest         |           3 |
| Staff         |          13 |
| Student       |           7 |
| Walkin        |           1 |
+---------------+-------------+
5 rows in set (0.0782 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL >
```

Figure 15.4: MySQL Most Common Visitor View

-Alan Verdin:

My contributions to the Front Desk Interface were several but the main was the Report's Menu. Where I added my teammates Views as well as my views and my procedure which acted like a filter. I added an All-Request Report View, All Deleted Request View. I also added the filter by visitor type view which uses my procedure to only print the desired visitor type requests.

Display Report Menu:

Figure 15.5: Interface Report Menu

Show All Request View:

VIEW ALL REQUEST REPORT

All Requests Shown Below:

Request ID	Request Type	Vistor Type	Request Status	Request Description
1409	Phone	Faculty	In Progress	Payroll Issues
2344	Walkin	Guest	Pending	Parking Ticket Issue
2406	Phone	Staff	In Progress	Wifi Issues
2489	Phone	Faculty	In Progress	Drop Student
2538	Email	Staff	In Progress	Wifi Issues
2801	Email	Staff	Pending	Need Hold Removal
3091	Email	Staff	Pending	Need Advising
3409	Walkin	Student	Pending	Payroll Issues
3782	Phone	Staff	Pending	Drop Student
4567	Email	Staff	Pending	Need Advising
4802	Walkin	Student	Pending	Wifi Issues
5001	Email	Student	In Progress	Need Advising
5002	Phone	Student	Pending	Need Hold Removal
5003	Phone	Faculty	In Progress	Payroll Issues
5004	Walkin	Faculty	Pending	Payroll Issues

Figure 15.6: Interface All Request View

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select *from request;
+-------------+---------------+---------------+-------------+----------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription         |
+-------------+---------------+---------------+-------------+----------------------+
|        1409 | Phone         | Faculty       | In Progress | Payroll Issues       |
|        2344 | Walkin        | Guest         | Pending     | Parking Ticket Issue |
|        2406 | Phone         | Staff         | In Progress | Wifi Issues          |
|        2489 | Phone         | Faculty       | In Progress | Drop Student         |
|        2538 | Email         | Staff         | In Progress | Wifi Issues          |
|        2801 | Email         | Staff         | Pending     | Need Hold Removal    |
|        3091 | Email         | Staff         | Pending     | Need Advising        |
|        3409 | Walkin        | Student       | Pending     | Payroll Issues       |
|        3782 | Phone         | Staff         | Pending     | Drop Student         |
|        4567 | Email         | Staff         | Pending     | Need Advising        |
|        4802 | Walkin        | Student       | Pending     | Wifi Issues          |
|        5001 | Email         | Student       | In Progress | Need Advising        |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal    |
|        5003 | Phone         | Faculty       | In Progress | Payroll Issues       |
|        5004 | Walkin        | Faculty       | Pending     | Payroll Issues       |
|        5006 | Email         | Staff         | Closed      | Schedule Update      |
|        5007 | Walkin        | Staff         | Pending     | Need Tech Support    |
|        5008 | Phone         | Faculty       | In Progress | Needs Advising       |
|        5429 | Phone         | Staff         | Pending     | Payroll Issues       |
|        5433 | Walkin        | Guest         | Pending     | Need Advising        |
|        5665 | Email         | Student       | Pending     | Need Advising        |
|        5737 | Email         | Staff         | Pending     | Need Advising        |
|        5767 | Walkin        | Student       | Pending     | Need Advising        |
|        6333 | Walkin        | Student       | Pending     | Wifi Issues          |
|        6758 | Walkin        | Guest         | Pending     | Payroll Issues       |
|        7438 | Email         | Staff         | In Progress | Payroll Issues       |
|        7908 | Phone         | Staff         | Pending     | Drop Student         |
|        8399 | Phone         | Faculty       | In Progress | Payroll Issues       |
|        8759 | Email         | Staff         | Pending     | Need Advising        |
|        9222 | Phone         | Faculty       | In Progress | Printers Issues      |
+-------------+---------------+---------------+-------------+----------------------+
```

Figure 15.7: MySQL All Requests

Show All Deleted Request View:

VIEW ALL DELETED REQUEST REPORT

All Deleted Requests Shown Below:

Date Deleted	Time Deleted	Deleted By Staff ID	Deleted Request ID
2021-08-22	3:25 PM MT	4001	5010
2021-09-05	1:05 PM MT	4001	5009
2022-04-28	23:25:28	4003	327
2022-04-29	00:00:05	4003	5009
2022-04-29	00:00:23	4003	5009
2022-04-29	00:00:39	4003	5009
2022-04-29	00:05:26	4003	327
2022-04-29	00:18:55	4003	327
2022-04-29	00:19:22	4003	1612
2022-04-29	00:24:00	4003	5010
2022-04-30	21:23:15	4003	5009
2022-04-30	21:27:45	4003	5005
2022-04-30	21:34:32	4003	78955
2022-04-30	22:25:01	4002	563

Return To Report Menu

Figure 15.8: Interface All Deleted Requests

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select *from deletes;
+------------+------------+----------+------------+
| Ddate      | Dtime      | Sstaff_ID | Rrequest_id |
+------------+------------+----------+------------+
| 2021-08-22 | 3:25 PM MT |      4001 |        5010 |
| 2021-09-05 | 1:05 PM MT |      4001 |        5009 |
| 2022-04-28 | 23:25:28   |      4003 |         327 |
| 2022-04-29 | 00:00:05   |      4003 |        5009 |
| 2022-04-29 | 00:00:23   |      4003 |        5009 |
| 2022-04-29 | 00:00:39   |      4003 |        5009 |
| 2022-04-29 | 00:05:26   |      4003 |         327 |
| 2022-04-29 | 00:18:55   |      4003 |         327 |
| 2022-04-29 | 00:19:22   |      4003 |        1612 |
| 2022-04-29 | 00:24:00   |      4003 |        5010 |
| 2022-04-30 | 21:23:15   |      4003 |        5009 |
| 2022-04-30 | 21:27:45   |      4003 |        5005 |
| 2022-04-30 | 21:34:32   |      4003 |       78955 |
| 2022-04-30 | 22:25:01   |      4002 |         563 |
+------------+------------+----------+------------+
```

Figure 15.9: MySQL All Deleted Requests

Procedure – Filter Request by Visitor:

This procedure contains a security implementation in PHP. The code will only take in the visitor types as input into the query. If it fails to meet the condition it will simply print input error.

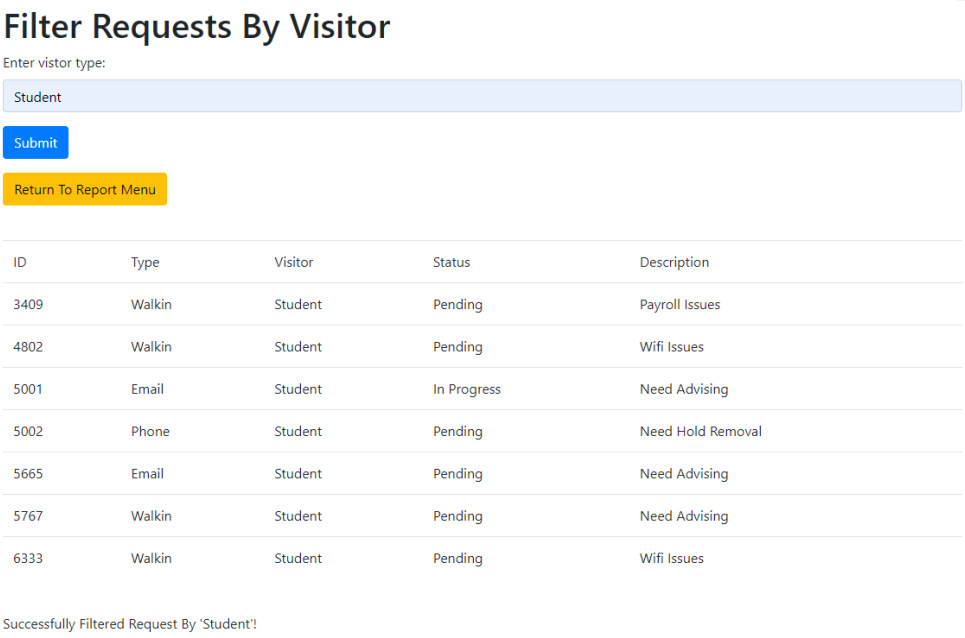
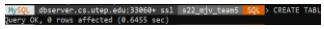


Figure 15.10 and 15.11: Interface Filter Requests By Visitor



```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > call getVisitorType('student');
+-------------+---------------+---------------+-------------+--------------------+
| Rrequest_id | Rrequest_type | Rvisitor_type | Rstatus     | Rdescription       |
+-------------+---------------+---------------+-------------+--------------------+
|        3409 | Walkin        | Student       | Pending     | Payroll Issues     |
|        4802 | Walkin        | Student       | Pending     | Wifi Issues        |
|        5001 | Email         | Student       | In Progress | Need Advising      |
|        5002 | Phone         | Student       | Pending     | Need Hold Removal  |
|        5665 | Email         | Student       | Pending     | Need Advising      |
|        5767 | Walkin        | Student       | Pending     | Need Advising      |
|        6333 | Walkin        | Student       | Pending     | Wifi Issues        |
+-------------+---------------+---------------+-------------+--------------------+
```

Figure 15.10 and 15.11: MySQL Filter Requests By Visitor

-Miguel: In the following images, the staff user will be able to tell which request type is the most common thanks to the Most Common Request Type Report, in this case, the most common request type is “Phone”.

Report for most common type of request

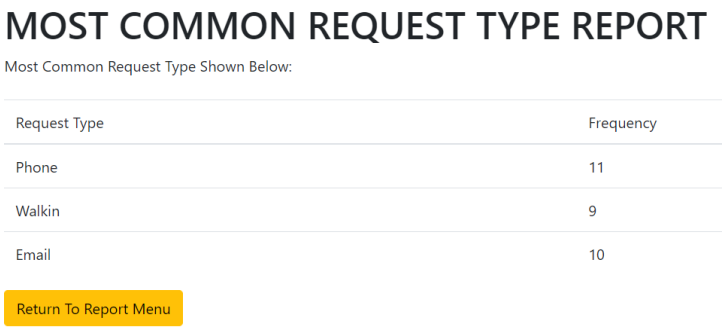


Figure 15.12: Interface Most Common Request Type

```
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL > select*from Number_Of_Requests;
+---------------+----------------+
| Rrequest_type | Rrequest_count |
+---------------+----------------+
| Phone         |             11 |
| Walkin        |              9 |
| Email         |             10 |
+---------------+----------------+
3 rows in set (0.0527 sec)
MySQL  dbserver.cs.utep.edu:33060+ ssl  s22_mjv_team5  SQL >
```

Figure 15.13: MySQL Most Common Request Type

-Garrett Jones:

My contribution to the interface was the ability to delete a request and store it in the deletes table.

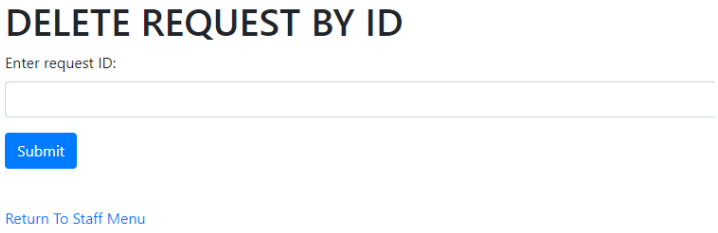


Figure 15.14: Delete Request Interface

Figure 15.15: Entering Request ID into Interface

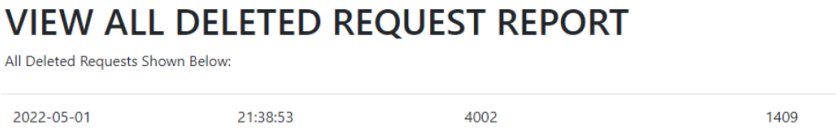


Figure 15.15: View All Deleted Request After Deletion

16. REQUIREMENTS TRACING

Requirement	Addressed By	Created By
R1. The system shall record the first name, last name, email and ID of each visitor.	Query – Figure 7.19	Alan Verdin

R2. The system shall assign a unique ID for each request.	Query – Figure 7.1	Christian Gomez
R3. The system shall allow visitors and staff to cancel a request.	Interface – Figure 17.3 & Figure 17.12	Christian Gomez
R4. The system will assign a pending status by default when a request is done by a visitor.	Interface – Figure 17.3	Christian Gomez
R5. The system will provide a list of options to declare if the request was made by email, phone or walkin.	Interface – Figure 17.3 & Figure 17.13	Christian Gomez
R6. Only staff members will be able to modify the status of the request, if needed.	Interface – Figure 17.14	Miguel Rodarte
R7. The system shall allow staff members to change any information from the request.	Interface – Figure 17.14	Miguel Rodarte
R8. The system shall allow assistant personnel to generate reports on the most common requests.	View – 13.1 Report – 15.1	Christian Gomez
R8. The system shall allow assistant personnel to generate reports of the most common type of requests.	View – 13.8 Report – 15.12	Miguel Rodarte
R8. The system shall allow assistant personnel to generate reports on the number of requests per day, week, month, year, semester	View – Figure 13.3-13.7	Garrett Jones
R8. The system shall allow assistant personnel to generate reports on the most common type of visitors.	Report – Figures 15.3 & 15.4	Christian Gomez
R8. The system shall allow assistant personnel to generate a report with all the requests.	Interface – Figure 17.6	Alan Verdin
R8. The system shall allow assistant personnel to generate a report with all deleted requests.	Interface – Figure 17.7 Procedure – Figure 14.6	Alan Verdin Garrett Jones
R8. The system shall allow assistant personnel to generate a report of all requests done by the type of visitor.	Interface – Figure 17.8	Alan Verdin

R8. The system shall allow assistant personnel to provide a report's menu.	Interface – Figure 17.5	Alan Verdin
R9. The system shall allow the assistant attending a request to change the request's status from pending to 'in progress', 'cancel', or 'done'.	Interface – Figure 17.14	Miguel Rodarte
R10. The system will provide a user-friendly interface where the visitor/staff can find and create requests easily. This will be implemented by adding color buttons and clear text messages.	Interface – Figures 17.1, 17.2, 17.3 and 17.13	Alan Verdin, Christian Gomez, Miguel Rodarte, and Garrett Jones

17. GRAPHICAL USER INTERFACE

- Figures 17.1 to 17.15 show the final GUI implementation for our Front Desk Project. Visitors will be able to create requests in a personalized menu. Within this menu the visitor will have to put their ID, first name, last name, type of visitor, and description of their problem. Also, the visitor will be able to cancel the request through the return to menu button. On the other hand, staff members will have a wide menu where they can create requests with more options to choose the type of information that the request will carry, have their own report menu, and menus where they can update and delete requests that are within the system. Within the Report's Menu, the staff will be able to generate or check reports that indicate the most common requests in their description, the most common request types, the number of requests received per week, all available requests depending on their type of visitor, and the most common types of visitors within requests.

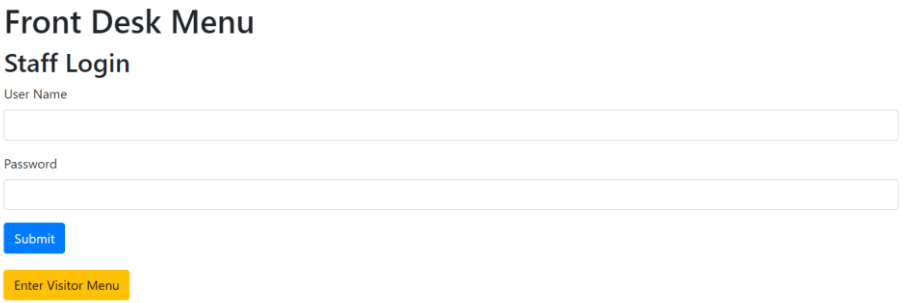


Figure 17.1: Main Menu

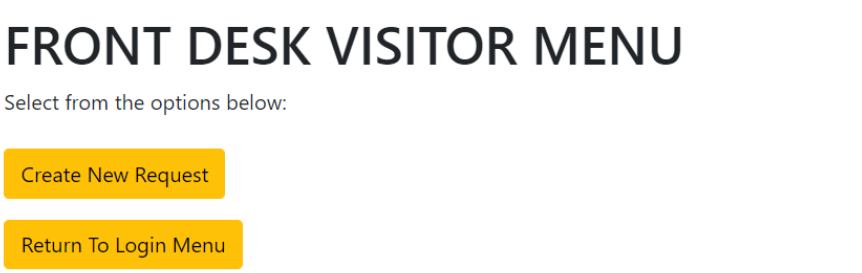


Figure 17.2: Visitor Menu

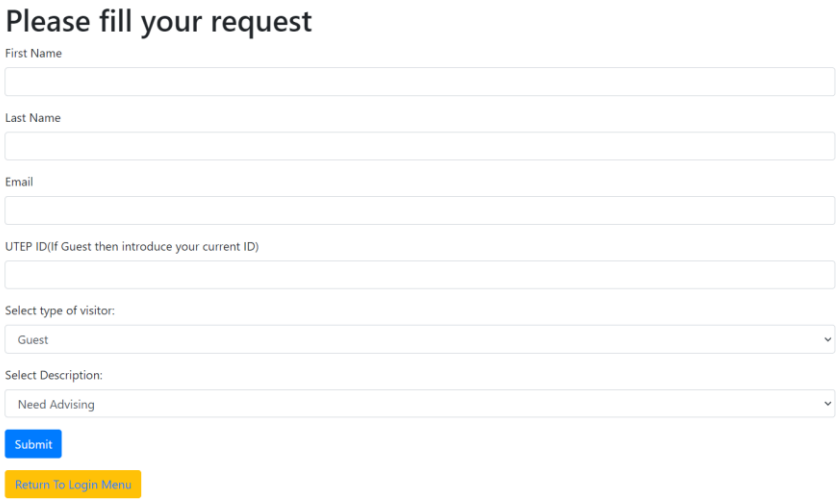


Figure 17.3: Create Request by Visitor

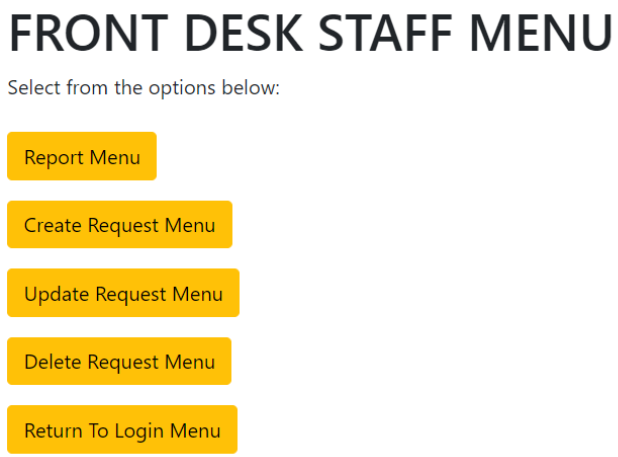


Figure 17.4: Front Desk Staff Menu

FRONT DESK REPORT MENU

Select from the report options below:

View All Requests

View All Deleted Requests

Filter Request By Vistor

View Number of Request Per Week

Most Common Request Description

Most Common Request Type

Most Common Visitor

Return To Staff Menu

Figure 17.5: Staff Report's Menu

VIEW ALL REQUEST REPORT

All Requests Shown Below:

Request ID	Request Type	Vistor Type	Request Status	Request Description
12	Faculty	Walkin	Pending	Need Advising
1409	Phone	Faculty	In Progress	Payroll Issues
2344	Walkin	Guest	Pending	Parking Ticket Issue
2406	Phone	Staff	In Progress	Wifi Issues
2489	Phone	Faculty	In Progress	Drop Student
2538	Email	Staff	In Progress	Wifi Issues
2801	Email	Staff	Pending	Need Hold Removal
3091	Email	Staff	Pending	Need Advising
3409	Walkin	Student	Pending	Payroll Issues
3782	Phone	Staff	Pending	Drop Student
4567	Email	Staff	Pending	Need Advising
4802	Walkin	Student	Pending	Wifi Issues

Figure 17.6: View All Requests from the system

VIEW ALL DELETED REQUEST REPORT

All Deleted Requests Shown Below:

Date Deleted	Time Deleted	Deleted By Staff ID	Deleted Request ID
2021-08-22	3:25 PM MT	4001	5010
2021-09-05	1:05 PM MT	4001	5009
2022-04-28	23:25:28	4003	327
2022-04-29	00:00:05	4003	5009
2022-04-29	00:00:23	4003	5009
2022-04-29	00:00:39	4003	5009
2022-04-29	00:05:26	4003	327
2022-04-29	00:18:55	4003	327
2022-04-29	00:19:22	4003	1612
2022-04-29	00:24:00	4003	5010
2022-04-30	21:23:15	4003	5009
2022-04-30	21:27:45	4003	5005

Figure 17.7: All Deleted Requests by Staff

Filter Requests By Visitor

Enter vistor type:

Submit

Return To Report Menu

Figure 17.8: Select all requests done by a type of visitor.

REQUEST PER WEEK REPORT

Weekly Requests Shown Below:

Year	Week	Frequency
		23
2022	6	1
2022	9	2
2022	10	1
2022	12	1
2022	15	2
2022	18	1
2002	20	2
2021	34	1

Return To Report Menu

Figure 17.9: Report of requests per week

MOST COMMON REQUEST DESCRIPTION REPORT

Most Common Request Description Shown Below:

Request Description	Frequency
Need Advising	9
Payroll Issues	8
Parking Ticket Issue	1
Wifi Issues	4
Drop Student	3
Need Hold Removal	2
Schedule Update	1
Need Tech Support	1
Needs Advising	1
Printers Issues	1

Return To Report Menu

Figure 17.10: Report of Most Common Requests by their description

MOST COMMON REQUEST TYPE REPORT

Most Common Request Type Shown Below:

Request Type	Frequency
Faculty	1
Phone	11
Walkin	9
Email	10

Return To Report Menu

Figure 17.11: Most Common type of Request

MOST COMMON VISTOR REPORT

Most Common Vistors Shown Below:

Visitor Type	Frequency
Faculty	7
Guest	3
Staff	13
Student	7
Walkin	1

Return To Report Menu

Figure 17.12: Report Most Common Visitors

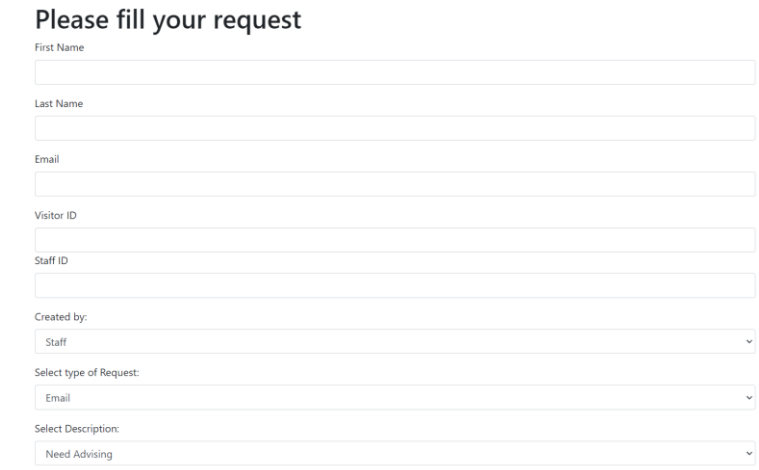


Figure 17.13: Create Request by Staff Menu

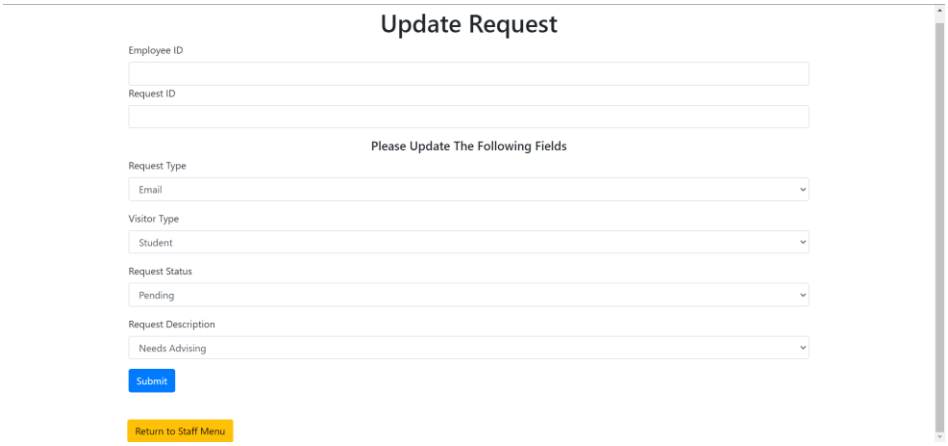


Figure 17.14: Update Request by Staff Menu

Figure 17.15: Delete Request by Staff Menu

18. IIS Web Server and GUI

-You can access the Staff Menu with the 2 following credentials in figure 18.2.

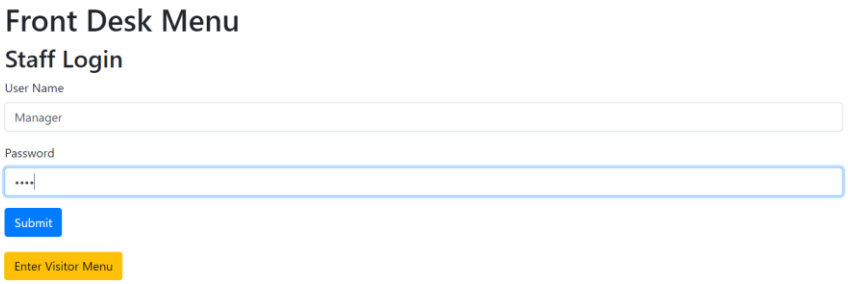


Figure 18.1: Where to input credentials

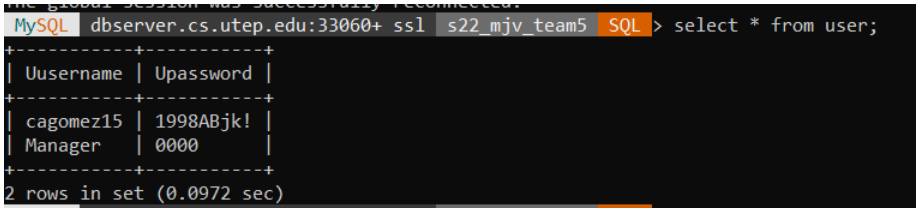


Figure 18.2: Credentials

19. REFERENCES

[1] Lecture 8: E/R Diagrams to Relational Model. Accessed on 2/15/2022.

[2] Converting ER Diagram to Schema | SQL | Tutorial 23. Accessed on 2/15/2022.

[3] Fundamentals of Database Systems (Book)

[4] https://www.guru99.com/er-diagram-tutorial-dbms.html

[5] https://cacoo.com/blog/er-diagrams-vs-eer-diagrams-whats-the-difference/

[6] https://www.youtube.com/watch?v=xQRRf5fOAt8

[7] https://www.youtube.com/watch?v=UrYLYV7WSHM&t=625s

[8] https://www.youtube.com/watch?v=Ipr9ws2bPEU

[9] https://www.youtube.com/watch?v=Z_hEj2U_5tI

[10] https://www.youtube.com/watch?v=9Pzj7Aj25lw

[11] https://www.youtube.com/watch?v=JTDK6r1GuUU

APPENDIX A. ATTRIBUTION INFORMATION

Christian Alberto Gomez:

Assignment 1:

1.1 Contributed to the third paragraph of the scope describing the Interface of our database. Also, I provided feedback to Alan during the scope. Finally. I provided a description of the request entity in the scope.

1.2 Contributed with 3 requirements. These requirements are focused on ID's.

1.3 Contributed to the Request section of the ER Diagram. Attributes such as Employee_ID and Employee_ID_Enter_Ticket were added by my teammate Garret Jones. Also, I provided a relationship between the visitor and the request.

1.4 Provided feedback for Employee and Visitor attributes.

1.5 Helped modeling and connection the Relational Model Schema.

1.6 Provided most of the journals from our meetings.

Assignment 2:

1.1 Provide help to fix Scope, Requirements and Assumptions.

1.2 Provided feedback and sources to fix our Relational Schema and Normalization.

1.3 Provided feedback and fixed our ER Diagram.

1.4 Created Visitor entity, visitor's attributes and provided feedback for all the relationships from the ER diagram.

1.5 Created and provided feedback for Request and Request_destination from our Relational Schema.

1.6 Provided feedback and help for the processing of normalization.

1.7 Created Request and Staff tables from MySQL Server.

1.8 Provided records for Request and Staff tables from MySQL Server.

1.9 Provided solutions to queries: “Get the most common types of requests” and “Get the total number of requests per category” from our database system.

1.10 Provided feedback on the other queries.

1.11 Completed parts 10 and 11 from assignment 2.

1.12 Provided feedback to my partners about how to put their information in part 10 and 11 from assignment 2.

Assignment 3:

1.1 Created and designed the bases of the PHP files of our project.

1.2 Fixed many comments or feedback from the requirements.

1.3 Provided two views as reports for the project.

1.4 Provided three procedures for the project.

1.5 Fixed ER Diagram according to the feedback.

1.6 Provided feedback and knowledge to my team members about how to edit and use the PHP files for our project.

1.7 Provided a conceptual map of how the project structure will be. This was during class.

1.8 Provided ideas and feedback to Alan for the creation of menus such as Staff Menu and Report’s Menu.

1.9 Designed the orange buttons for all interfaces.

1.10 Most screen shots from assignment 3 with their figure description were added by me. Garret screenshots and his figures descriptions are from him.

1.11 Created and filled REQUIREMENTS TRACING Table.

1.12 Helped to fix bugs or errors from Miguel’s PHP files.

1.13 Provided ideas to Miguel about how to design his Update Menu.

1.14 Tested frequently the website, PHP files, and MySQL tables to check if everything was working correctly.

1.15 Connected our project to the team database.

1.16 Added parameter for creating requests to select the correct options according to our project.

1.17 Wrote descriptions, figures, and took all screen shots for GUI section.

1.18 Wrote and took pictures for ISS Webserver and GUI section.

Garret William Jones:

2. Garrett William Jones

2.1 Assisted with paragraphs' 2, 4, and 6 of the scope. Specifically regarding the database audience, communication with the visitor, and types' of information collected.

2.2 added solicitation type and status modification requirements.

2.3 Contributed service's and employee's relationships to request.

2.4 Contributed Service, employee, and employee type tables in relational schema.

2.5 Provided feedback related to ER diagram and relational schema to teammates.

Assignment 3:

2.6 Wrote views to return number of requests per day, week, month, year, and semester

2.7 Created a procedure DeleteByRequest_ID which deletes a request from the request table and stores related info in the deletes table.

2.8 Contributed php file in interface which utilizes the DeleteByRequest_ID procedure.

2.9 Worked with Alan to ensure that interface inputs were valid

3.0 Brainstormed security measures with team to protect against SQL injection and other attacks.

3.Miguel Rodarte

Assignment 1:

3.1 Contributed with the scope, more specifically with what the system should be able take from the visitor.

3.2 Provided suggestions for the assumptions of our project.

3.3 Contributed with 3 requirements of the system, forward, keep count, and filter requests.

3.4 Contributed with the creation of Visitor entity in the E/R diagram.

3.5 Contributed with the first design of the Relational Model and worked together with teammates to revise into a new Relational Model.

Assignment 2:

3.1 Revised the scope with feedback given to us by the instructor.

3.2 Assisted with new requirements and assumptions.

3.3 Created tables for creates and request_destination.

3.4 Made insert statements for deletes and creates tables.

3.5 Worked on SQL queries and showed the requirements needed.

3.6 Successfully completed sections 10 & 11 with the help of my teammates.

Assignment 3:

3.7 Created a view for how many requests are done by email, phone, and Walkin.

3.8 Created procedure to update request table.

3.9 Created procedure to update updates table.

3.10 Created interface for the implementation of updating a request after teammate Christian thoroughly explained the file during class.

3.11 Received feedback from teammates on how to fix bugs in my files and fixed them accordingly.

3.12 Added security feature to updating request interface to allow only employees with an ID to update a request.

4. Alan Verdin

Assignment 1:

4.1 As a team we annotated and discussed the requirements of the project to ensure that the scope is correct.

4.2 Contributed with Christian to create the final draft of the scope. While keeping track of the requirements.

4.3 Contributed with the entire team to design the first few attempts of the E/R Diagram. Later finalized the E/R diagram with Christian and Garret.

4.4 Discussed with the TA and the rest of the team on utilizing a disjoint entity, “person”.

4.5 Assisted with revising the first design of the Relational Model Miguel made.

4.6 Assisted in making the final Relational Model with Garret, Christian.

Assignment 2:

4.7 Worked with the team to design a better ER diagram, to move on to the properly make a relational diagram and the normalization process.

4.8 Used the schema to properly split and separate the tables as required to complete the normalization process. To ensure all tables in the normalization were correct I had Christian's assistance to ensure the final normalization was correct.

4.9 Garret and I found some useful date management commands for SQL to facilitate some of the queries. I created a view that organized all dates and requests into one table to create queries from the table.

4.10 I created the index file for team 5 to allow the directories of my team to be found in the team 5 server section.

4.11 Assisted my team alongside Christian to ensure their GUI was properly working.

4.12 I designed most of the hypothetical test data used to fill the data tables for our project. The data will soon grow to allow better queries and examples of use. Once designed the entire team contributed to creating and filling the tables in SQL.

4.13 Completed two of the queries for the assignment and assisted in completing the others.

Assignment 3:

4.14 After Christian gave me a rundown of how the PHP worked in relation to the Gui. Everything became easier for me and him to create the bases of the logins and menus for the interface. Which allowed us to explain to the rest of our team.

4.15 Christian and I made the Login Menu Interface, Visitor and Staff Menu.

4.16 I made the Report Menu that holds all of the team's views and my procedure as it is used a filter for the requests.

4.17 I created the Filter Request by Visitor view that uses my procedure, to print only the request with the wanted visitor type.

4.18 I also created the views to print all requests and deleted requests in the report menu.

4.19 I also assisted my team to solve bugs and issues they had with their code and procedures to reach the desired outcome.

4.20 Added input security parameters to the menu Filter By Visitor which will only take visitor types and the menu to delete requests which will only take numeric input.

IN YOUR TEAM JOURNAL

Each team member should reply to the entry saying: I agree with the contribution of my teammates stated in this journal entry.

Christian Gomez: I agree with the contribution of my teammates stated in this journal entry.

Alan Verdin: I agree with the contribution of my teammates stated in this journal entry.

Garrett Jones: I agree with the contribution of my teammates stated in this journal entry.

Miguel Rodarte: I agree with the contribution of my teammates stated in this journal entry.


MySQL dbserver.cs.utep.edu:33060+ ssl s22_mjv_team SQL > CREATE TABL
Query OK, 0 rows affected (0.6455 sec)